

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



**BÀI TIỂU LUẬN
KIẾN TRÚC VÀ THIẾT KẾ PHẦN MỀM**

Sinh Viên: TRẦN MINH QUANG

Lớp: CNPM05

Mã Sinh Viên: B20DCCN537

Hà Nội, 2024

Câu 1: Mô tả bằng ngôn ngữ tự nhiên các chức năng tương ứng với các actor và phi chức năng của Hệ thống ecomSys.

Hệ thống ecomSys, gồm các chức năng chính sau:

- *Khách hàng:*
 - Đăng ký, đăng nhập, khôi phục mật khẩu, cập nhật thông tin cá nhân.
 - Tìm kiếm và xem chi tiết sản phẩm, thêm vào giỏ hàng, đặt hàng, hủy đơn hàng, thanh toán.
- *User/Manager:*
 - Đăng ký, đăng nhập, khôi phục mật khẩu, cập nhật thông tin cá nhân.
 - Quản lý sản phẩm (thêm, sửa, xóa, xem chi tiết), quản lý khách hàng (thêm, sửa, xóa, xem chi tiết), quản lý đơn hàng (xem chi tiết, xác nhận, hủy), xem thống kê (doanh thu, số lượng sản phẩm tồn kho, ...).
- *Yêu cầu phi chức năng:*
 - Bảo trì: Hệ thống cần có thời gian bảo trì ngắn.
 - Bảo mật: Bảo mật thông tin của khách hàng là ưu tiên hàng đầu.
 - Hiệu suất: Đảm bảo hiệu suất cao trong điều kiện có lượng lớn truy cập vào hệ thống.
 - Độ tin cậy: Hệ thống phải có khả năng sao lưu dữ liệu để đề phòng mất mát thông tin.

MÔ TẢ CHỨC NĂNG BẰNG NGÔN NGỮ TỰ NHIÊN

Actor: Khách hàng_Customer.

- Hoạt động chính:
 - Đăng ký tài khoản và đăng nhập trên hệ thống.
 - Xem sản phẩm.
 - Tìm kiếm.
 - Đặt hàng.
 - Thanh toán.
 - Theo dõi hoặc hủy đơn hàng.
 - Thực hiện đánh giá đơn hàng.

Actor: Nhân viên_User.

- Hoạt động chính:
 - Đăng nhập.

Xác nhận đơn hàng.
Thêm, sửa, xóa.
Xác nhận hủy đơn.
Xem đánh giá của khách hàng.

MÔ TẢ CÁC CHỨC NĂNG

1. Khách hàng đăng ký tài khoản:

- Khách hàng chọn chức năng đăng ký tài khoản trên trang chủ của hệ thống → Giao diện đăng ký tài khoản hiện ra với các ô nhập thông tin (tên, số điện thoại, email, mật khẩu) và nút Đăng ký → Khách hàng điền tất cả các thông tin vào các ô sau đó nhấn Đăng ký → Giao diện thông báo đăng ký thành công và quay trở về trang chủ của hệ thống.

2. Khách hàng đăng nhập:

- Khách hàng chọn chức năng đăng nhập tài khoản trên trang chủ của hệ thống → Giao diện đăng nhập hiện ra với các ô nhập thông tin khách hàng (tên, mật khẩu) và nút Đăng nhập và nút Quên mật khẩu → Khách hàng điền thông tin tên đăng nhập, mật khẩu và nhấn nút Đăng nhập → Giao diện hiện thông báo đăng nhập thành công (nếu tên đăng nhập, mật khẩu chính xác) và hiện ra giao diện trang chủ có các sản phẩm hoặc thông báo đăng nhập thất bại (nếu tên đăng nhập, mật khẩu không chính xác) và giao diện đăng nhập hiện ra để người dùng đăng nhập lại.

3. Khách hàng quên mật khẩu:

- Khách hàng chọn chức năng đăng nhập tài khoản trên trang chủ của hệ thống → Giao diện đăng nhập hiện ra với các ô nhập thông tin khách hàng (tên, mật khẩu) và nút Đăng nhập và nút Quên mật khẩu → Khách hàng chọn nút Quên mật khẩu → Giao diện lấy lại mật khẩu hiện ra với ô nhập tên và email khôi phục và nút Tiếp theo → Khách hàng nhập thông tin tên đăng nhập và email khôi phục và nhấn nút Tiếp theo → Giao diện hiện ra ô nhập mã xác nhận được gửi về email khôi phục và nút Xác nhận → Khách hàng nhập mã xác nhận và click vào nút Xác nhận → Giao diện hiện ra với ô nhập mật khẩu mới và nút Xác nhận → Khách hàng nhập mật khẩu mới và nút xác nhận → Giao diện hiện ra thông báo thay đổi mật khẩu thành công và quay trở về trang chủ của hệ thống.

4. Khách hàng thay đổi thông tin cá nhân:

- Khách hàng đăng nhập vào hệ thống → Giao diện trang chủ của khách hàng hiện ra → Khách hàng chọn thay đổi thông tin cá nhân → Giao diện hiện ra với các ô chứa thông tin khách hàng (tên, số điện thoại, email, mật khẩu) và nút Cập nhật → Khách hàng điền vào các ô mà mình muốn thay đổi và nhấn Cập nhật → Giao diện hiện thông báo Cập nhật thông tin thành công và quay trở về trang chủ của khách hàng.

5. Khách hàng tạo giỏ hàng:

- Khách hàng đăng nhập hệ thống → Giao diện trang chủ của khách hàng hiện ra với ô nhập tên sản phẩm tìm kiếm và nút Tìm kiếm → Khách hàng nhập tên sản phẩm muốn tìm kiếm → Giao diện hiện ra các sản phẩm có tên chứa ký tự mà khách hàng muốn tìm kiếm (cạnh mỗi sản phẩm có nút Thêm giỏ hàng) → Khách hàng click Thêm giỏ hàng → Giao diện thêm giỏ hàng hiện ra với thông tin sản phẩm (ảnh mô tả, tên sản phẩm, tác giả, giá, giá được giảm, mô tả, loại hàng, số lượng), ô nhập số lượng sản phẩm và nút Thêm giỏ hàng → Khách hàng nhập số lượng sản phẩm cần mua, click Thêm giỏ hàng → Giao diện giỏ hàng hiện ra với sản phẩm khách hàng vừa thêm kèm số lượng sản phẩm.

6. Khách hàng đặt hàng:

- Đặt hàng không tìm kiếm: Khách hàng đăng nhập hệ thống → Giao diện trang chủ của khách hàng hiện ra → Khách hàng click vào giỏ hàng → Giao diện giỏ hàng hiện ra với các sản phẩm đã được thêm trước đó (có nút checkbox ở bên trái) cùng ô số lượng đã thêm trước đó → Khách hàng tích vào sản phẩm mình cần mua, thay đổi số lượng sản phẩm (nếu cần) → Giao diện hiện ra số tiền phải thanh toán và nút Mua → Khách hàng xác nhận và click Mua → Giao diện hiện ra thông tin đơn hàng (sản phẩm, số lượng, số tiền thanh toán) và ô chọn nhà vận chuyển, chọn hình thức thanh toán (tiền mặt hoặc online) và nút Xác nhận → Khách hàng chọn đơn vị vận chuyển, hình thức thanh toán và click nút Xác nhận → Giao diện hiện ra thông tin chi tiết đơn hàng (tên sản phẩm, số lượng sản phẩm, đơn vị vận chuyển, hình thức thanh toán, tổng tiền cần trả) và nút Đặt hàng → Khách hàng xác nhận và click Đặt hàng → Giao diện thông báo đơn hàng được đặt thành công và quay trở về trang chủ của khách hàng.

7. Đặt hàng có tìm kiếm:

- Khách hàng đăng nhập hệ thống → Giao diện trang chủ của khách hàng hiện ra với ô nhập tên sản phẩm tìm kiếm và nút Tìm kiếm → Khách hàng nhập tên sản phẩm muốn tìm kiếm → Giao diện hiện ra các sản phẩm có tên chứa ký tự mà khách hàng muốn tìm kiếm (cạnh mỗi sản phẩm có nút Thêm giỏ hàng) → Khách hàng click Thêm giỏ hàng → Giao diện thêm giỏ hàng hiện ra với thông tin sản phẩm (ảnh mô tả, tên sản phẩm, tác giả, giá, giá được giảm, mô tả, loại hàng, số lượng), ô nhập số lượng sản phẩm và nút Thêm giỏ hàng → Khách hàng nhập số lượng sản phẩm cần mua, click Thêm giỏ hàng → → Giao diện giỏ hàng hiện ra với các sản phẩm đã được thêm trước đó (có nút checkbox ở bên trái) cùng ô số lượng đã thêm trước đó → Khách hàng tích vào sản phẩm mình cần mua, thay đổi số lượng sản

phẩm (nếu cần) → Giao diện hiện ra số tiền phải thanh toán và nút Mua → Khách hàng xác nhận và click Mua → Giao diện hiện ra thông tin đơn hàng (sản phẩm, số lượng, số tiền thanh toán) và ô chọn nhà vận chuyển, chọn hình thức thanh toán (tiền mặt hoặc online) và nút Xác nhận → Khách hàng chọn đơn vị vận chuyển, hình thức thanh toán và click nút Xác nhận → Giao diện hiện ra thông tin chi tiết đơn hàng (tên sản phẩm, số lượng sản phẩm, đơn vị vận chuyển, hình thức thanh toán, tổng tiền cần trả) và nút Đặt hàng → Khách hàng xác nhận và click Đặt hàng → Giao diện thông báo đơn hàng được đặt thành công và quay trở về trang chủ của khách hàng.

8. Khách hàng huỷ đơn hàng:

- Khách hàng đăng nhập vào hệ thống → Giao diện trang chủ của khách hàng hiện ra → Khách hàng vào mục đơn hàng của bạn → Giao diện hiện ra thông tin của đơn hàng và trạng thái của đơn hàng, nếu đơn hàng chưa vận chuyển thì có thể click nút Hủy đơn hàng sáng lên, đơn hàng đã vận chuyển thì nút Hủy đơn hàng tối lại và không thể click được → Khách hàng click Hủy đơn hàng → Giao diện hiện ra với thông báo Bạn có chắc chắn huỷ đơn hàng và nút Xác nhận → Khách hàng click nút Xác nhận → Thông báo huỷ đơn hàng thành công và quay trở về trang chủ của khách hàng.

9. Nhân viên đăng ký tài khoản:

- Nhân viên chọn chức năng đăng ký tài khoản trên trang chủ của hệ thống → Giao diện đăng ký tài khoản hiện ra với các ô nhập thông tin (tên, số điện thoại, email, mật khẩu) và nút Đăng ký → Nhân viên điền tất cả các thông tin vào các ô sau đó nhấn Đăng ký → Giao diện thông báo đăng ký thành công và quay trở về trang chủ của hệ thống.

10. Nhân viên đăng nhập:

- Nhân viên chọn chức năng đăng nhập tài khoản trên trang chủ của hệ thống → Giao diện đăng nhập hiện ra với các ô nhập thông tin nhân viên (tên, mật khẩu) và nút Đăng nhập và nút Quên mật khẩu → Nhân viên điền thông tin tên đăng nhập, mật khẩu và nhấn nút Đăng nhập → Giao diện hiện thông báo đăng nhập thành công (nếu tên đăng nhập, mật khẩu chính xác) và hiện ra giao diện trang chủ có các sản phẩm hoặc thông báo đăng nhập thất bại (nếu tên đăng nhập, mật khẩu không chính xác) và giao diện đăng nhập hiện ra để người dùng đăng nhập lại.

11. Nhân viên quên mật khẩu:

- Nhân viên chọn chức năng đăng nhập tài khoản trên trang chủ của hệ thống → Giao diện đăng nhập hiện ra với các ô nhập thông tin nhân viên (tên, mật

khẩu) và nút Đăng nhập và nút Quên mật khẩu → Nhân viên chọn nút Quên mật khẩu → Giao diện lấy lại mật khẩu hiện ra với ô nhập tên và email khôi phục và nút Tiếp theo → Khách hàng nhập thông tin tên đăng nhập và email khôi phục và nhấn nút Tiếp theo → Giao diện hiện ra ô nhập mã xác nhận được gửi về email khôi phục và nút Xác nhận → Nhân viên nhập mã xác nhận và click vào nút Xác nhận → Giao diện hiện ra với ô nhập mật khẩu mới và nút Xác nhận → Nhân viên nhập mật khẩu mới và nút xác nhận → Giao diện hiện ra thông báo thay đổi mật khẩu thành công và quay trở về trang chủ của hệ thống.

12.Nhân viên thay đổi thông tin cá nhân:

- Nhân viên đăng nhập vào hệ thống → Giao diện trang chủ của nhân viên hiện ra → Nhân viên chọn thay đổi thông tin cá nhân → Giao diện hiện ra với các ô chứa thông tin nhân viên (tên, số điện thoại, email, mật khẩu) và nút Cập nhật → Nhân viên điền vào các ô mà mình muốn thay đổi và nhấn Cập nhật → Giao diện hiện thông báo Cập nhật thông tin thành công và quay trở về trang chủ của nhân viên.

13.Nhân viên quản lý sản phẩm:

- Thêm sản phẩm: Nhân viên đăng nhập vào hệ thống → Giao diện trang chủ của nhân viên hiện ra → Nhân viên click Thêm sản phẩm → Giao diện thêm sản phẩm hiện ra với các ô ảnh mô tả, tên sản phẩm, tác giả, giá, giá được giảm, mô tả, loại hàng, số lượng và nút Thêm sản phẩm → Nhân viên điền các thông tin về sản phẩm vào các ô (không để trống ô nào) và click vào nút Thêm sản phẩm → Thông báo thêm sản phẩm thành công và quay trở lại trang chủ của nhân viên.

14.Sửa sản phẩm:

- Nhân viên đăng nhập vào hệ thống → Giao diện trang chủ của nhân viên hiện ra cùng ô tìm kiếm và nút Tìm kiếm → Nhân viên nhập tên sản phẩm muốn tìm → Giao diện hiện ra danh sách các sản phẩm có chứa ký tự mà nhân viên nhập gồm ảnh mô tả sản phẩm, tên sản phẩm và nút Cập nhật, nút Xóa, Xem chi tiết → Nhân viên chọn nút Cập nhật → Giao diện hiện ra với các ô về thông tin chi tiết của sản phẩm (ảnh mô tả, tên sản phẩm, tác giả, giá, giá được giảm, mô tả, loại hàng, số lượng) và nút Cập nhật → Nhân viên điền các ô muốn sửa thông tin và nhấn Cập nhật → Giao diện thông báo cập nhật thành công và quay trở về trang chủ của nhân viên.

15.Xóa sản phẩm:

- Nhân viên đăng nhập vào hệ thống → Giao diện trang chủ của nhân viên hiện ra cùng ô tìm kiếm và nút Tìm kiếm → Nhân viên nhập tên sản phẩm muốn tìm → Giao diện hiện ra danh sách các sản phẩm có chứa ký tự mà nhân viên nhập gồm ảnh mô tả sản phẩm, tên sản phẩm và nút Cập nhật, nút Xóa, Xem chi tiết → Nhân viên chọn nút Xóa → Giao diện hiện thị thông báo Bạn có chắc muốn xóa sản phẩm này không? kèm nút Có, nút

Không → Nhân viên click Có → Thông báo xoá thành công và quay lại trang chủ của nhân viên.

16. Xem chi tiết sản phẩm:

- Nhân viên đăng nhập vào hệ thống → Giao diện trang chủ của nhân viên hiện ra cùng ô tìm kiếm và nút Tìm kiếm → Nhân viên nhập tên sản phẩm muốn tìm → Giao diện hiện ra danh sách các sản phẩm có chứa ký tự mà nhân viên nhập gồm ảnh mô tả sản phẩm, tên sản phẩm và nút Cập nhật, nút Xoá, Xem chi tiết → Nhân viên chọn nút Xem chi tiết → Giao diện hiện ra với các ô về thông tin chi tiết của sản phẩm (ảnh mô tả, tên sản phẩm, tác giả, giá, giá được giảm, mô tả, loại hàng, số lượng) và nút Xoá, nút Cập nhật, nút Thoát → Nhân viên click Thoát → Quay trở về trang chủ của nhân viên.

17. Nhân viên quản lý khách hàng:

Thêm khách hàng: Nhân viên đăng nhập vào hệ thống → Giao diện trang chủ của nhân viên hiện ra → Nhân viên click Thêm khách hàng → Giao diện thêm khách hàng hiện ra với các ô tên, số điện thoại, email, mật khẩu và nút Thêm khách hàng → Nhân viên điền các thông tin về khách hàng vào các ô (không để trống ô nào) và click vào nút Thêm khách hàng → Thông báo thêm khách hàng thành công và quay trở lại trang chủ của nhân viên.

Sửa khách hàng: Nhân viên đăng nhập vào hệ thống → Giao diện trang chủ của nhân viên hiện ra cùng ô tìm kiếm và nút Tìm kiếm → Nhân viên nhập tên khách hàng muốn tìm → Giao diện hiện ra danh sách các khách hàng có chứa ký tự mà nhân viên nhập gồm tên, số điện thoại, email, mật khẩu và nút Cập nhật, nút Xoá, Xem chi tiết → Nhân viên chọn nút Cập nhật → Giao diện hiện ra với các ô về thông tin chi tiết của khách hàng (tên, số điện thoại, email, mật khẩu) và nút Cập nhật → Nhân viên điền các ô muốn sửa thông tin và nhấn Cập nhật → Giao diện thông báo cập nhật thành công và quay trở về trang chủ của nhân viên.

Xoá khách hàng: Nhân viên đăng nhập vào hệ thống → Giao diện trang chủ của nhân viên hiện ra cùng ô tìm kiếm và nút Tìm kiếm → Nhân viên nhập tên khách hàng muốn tìm → Giao diện hiện ra danh sách các khách hàng có chứa ký tự mà nhân viên nhập gồm tên, số điện thoại, email, mật khẩu và nút Cập nhật, nút Xoá, Xem chi tiết → Nhân viên chọn nút Xoá → Giao diện hiện thị thông báo Bạn có chắc muốn xoá khách hàng này không? kèm nút Có, nút Không → Nhân viên click Có → Thông báo xoá thành công và quay lại trang chủ của nhân viên.

Xem chi tiết khách hàng: Nhân viên đăng nhập vào hệ thống → Giao diện trang chủ của nhân viên hiện ra cùng ô tìm kiếm và nút Tìm kiếm → Nhân viên nhập tên khách hàng muốn tìm → Giao diện hiện ra danh sách các khách hàng có chứa ký tự mà nhân viên nhập gồm tên, số điện thoại, email, mật khẩu và nút Cập nhật, nút Xoá, Xem chi tiết → Nhân viên chọn nút Xem chi tiết → Giao diện hiện ra

với các ô về thông tin chi tiết của khách hàng (tên, số điện thoại, email, mật khẩu) và nút Xóa, nút Cập nhật, nút Thoát → Nhân viên click Thoát → Quay trở về trang chủ của nhân viên.

18. Nhân viên quản lý đơn hàng

Xem chi tiết đơn hàng: Nhân viên đăng nhập vào hệ thống → Giao diện trang chủ của nhân viên hiện ra → Nhân viên click Đơn hàng → Giao diện đơn hàng hiện ra với danh sách các đơn hàng và ô tìm kiếm (theo tên khách hàng hoặc mã đơn hàng), nút Tìm kiếm → Nhân viên nhập tên khách hàng hoặc mã đơn hàng và click nút Tìm kiếm → Giao diện hiện ra với danh sách các đơn hàng và nút Xem chi tiết → Nhân viên click Xem chi tiết vào đơn hàng mình muốn xem → Giao diện đơn hàng hiện ra với thông tin mã đơn hàng, tên khách hàng, tên sản phẩm, số lượng, giá mỗi sản phẩm, tổng tiền thanh toán, đơn vị vận chuyển, hình thức thanh toán và nút Thoát → Nhân viên click Thoát → Giao diện quay về trang chủ của nhân viên.

Xác nhận đơn hàng: Nhân viên đăng nhập vào hệ thống → Giao diện của nhân viên hiện ra → Nhân viên chọn xác nhận đặt hàng → Giao diện hiện ra danh sách các đơn hàng đã đặt của khách hàng và nút Xác nhận, Huỷ bỏ → Nhân viên click nút xác nhận cho từng đơn hàng và cập nhật dữ liệu để khách hàng dễ dàng theo dõi vận chuyển đơn hàng của mình → Thông báo thành công và quay về trang chủ các đơn hàng cần xác nhận.

Huỷ bỏ đơn hàng: Nhân viên đăng nhập vào hệ thống → Giao diện của nhân viên hiện ra → Nhân viên chọn xác nhận đơn hàng → Giao diện hiện ra danh sách các đơn hàng đã đặt của khách hàng và nút Xác nhận, Huỷ bỏ → Nhân viên click nút Huỷ bỏ cho đơn hàng muốn huỷ và cập nhật dữ liệu để khách hàng dễ dàng theo dõi vận chuyển đơn hàng của mình → Thông báo thành công và quay về trang chủ các đơn hàng cần xác nhận.

19. Nhân viên xem thống kê doanh thu trong khoảng thời gian:

- Nhân viên đăng nhập vào hệ thống → Giao diện của nhân viên hiện ra → Nhân viên chọn Xem thống kê doanh thu → Giao diện hiện ra ô nhập từ ngày, đến ngày và nút Xem thống kê → Nhân viên nhập ngày bắt đầu, ngày kết thúc và click Xem thống kê → Giao diện hiện ra tổng doanh thu và danh sách các đơn hàng trong thời gian nhân viên muốn tìm kiếm.

MÔ TẢ CHỨC NĂNG:

Đăng kí

Usecase	Đăng ký
Actor	Khách hàng

Tiền điều kiện	
Hậu điều kiện	Đăng ký thành công
Hoạt động	Khách hàng chọn chức năng đăng ký tài khoản trên trang chủ của hệ thống → Giao diện đăng ký tài khoản hiện ra với các ô nhập thông tin (tên, số điện thoại, email, mật khẩu) và nút Đăng ký → Khách hàng điền tất cả các thông tin vào các ô sau đó nhấn Đăng ký → Giao diện thông báo đăng ký thành công và quay trở về trang chủ của hệ thống.

Đăng nhập

Usecase	Đăng nhập
Actor	Khách hàng
Tiền điều kiện	Tên đăng nhập, mật khẩu
Hậu điều kiện	Đăng nhập thành công
Hoạt động	Khách hàng chọn chức năng đăng nhập tài khoản trên trang chủ của hệ thống → Giao diện đăng nhập hiện ra với các ô nhập thông tin khách hàng (tên, mật khẩu) và nút Đăng nhập và nút Quên mật khẩu → Khách hàng điền thông tin tên đăng nhập, mật khẩu và nhấn nút Đăng nhập → Giao diện hiện thông báo đăng nhập thành công (nếu tên đăng nhập, mật khẩu chính xác) và hiện ra giao diện trang chủ có các sản phẩm hoặc thông báo đăng nhập thất bại (nếu tên đăng nhập, mật khẩu không chính xác) và giao diện đăng nhập hiện ra để người dùng đăng nhập lại.

Quên mật khẩu

Usecase	Quên mật khẩu
Actor	Khách hàng
Tiền điều kiện	Tên đăng nhập
Hậu điều kiện	Tạo mật khẩu mới thành công

Hoạt động	Khách hàng chọn chức năng đăng nhập tài khoản trên trang chủ của hệ thống → Giao diện đăng nhập hiện ra với các ô nhập thông tin khách hàng (tên, mật khẩu) và nút Đăng nhập và nút Quên mật khẩu → Khách hàng chọn nút Quên mật khẩu → Giao diện lấy lại mật khẩu hiện ra với ô nhập tên và email khôi phục và nút Tiếp theo → Khách hàng nhập thông tin tên đăng nhập và email khôi phục và nhấn nút Tiếp theo → Giao diện hiện ra ô nhập mã xác nhận được gửi về email khôi phục và nút Xác nhận → Khách hàng nhập mã xác nhận và click vào nút Xác nhận → Giao diện hiện ra với ô nhập mật khẩu mới và nút Xác nhận → Khách hàng nhập mật khẩu mới và nút xác nhận → Giao diện hiện ra thông báo thay đổi mật khẩu thành công và quay trở về trang chủ của hệ thống.
-----------	--

Thay đổi thông tin cá nhân

Usecase	Thay đổi thông tin cá nhân
Actor	Khách hàng
Tiền điều kiện	Tên đăng nhập, mật khẩu
Hậu điều kiện	Thay đổi thông tin thành công
Hoạt động	Khách hàng đăng nhập vào hệ thống → Giao diện trang chủ của khách hàng hiện ra → Khách hàng chọn thay đổi thông tin cá nhân → Giao diện hiện ra với các ô chứa thông tin khách hàng (tên, số điện thoại, email, mật khẩu)
	và nút Cập nhật → Khách hàng điền vào các ô mà mình muốn thay đổi và nhấn Cập nhật → Giao diện hiện thông báo Cập nhật thông tin thành công và quay trở về trang chủ của khách hàng.

Tạo cart

Usecase	Tạo giỏ hàng
Actor	Khách hàng
Tiền điều kiện	Tên đăng nhập, mật khẩu
Hậu điều kiện	Tạo giỏ hàng thành công

Hoạt động	Khách hàng đăng nhập hệ thống → Giao diện trang chủ của khách hàng hiện ra với ô nhập tên sản phẩm tìm kiếm và nút Tìm kiếm → Khách hàng nhập tên sản phẩm muốn tìm kiếm → Giao diện hiện ra các sản phẩm có tên chứa kí tự mà khách hàng muốn tìm kiếm (cạnh mỗi sản phẩm có nút Thêm giỏ hàng) → Khách hàng click Thêm giỏ hàng → Giao diện thêm giỏ hàng hiện ra với thông tin sản phẩm (ảnh mô tả, tên sản phẩm, tác giả, giá, giá được giảm, mô tả, loại hàng, số lượng), ô nhập số lượng sản phẩm và nút Thêm giỏ hàng → Khách hàng nhập số lượng sản phẩm cần mua, click Thêm giỏ hàng → Giao diện giỏ hàng hiện ra với sản phẩm khách hàng vừa thêm kèm số lượng sản phẩm.
-----------	--

Order

Usecase	Đặt hàng
Actor	Khách hàng
Tiền điều kiện	Tên đăng nhập, mật khẩu
Hậu điều kiện	Đặt hàng thành công
Hoạt động	Khách hàng đăng nhập hệ thống → Giao diện trang chủ của khách hàng hiện ra với ô nhập tên sản phẩm tìm kiếm và nút Tìm kiếm → Khách hàng nhập tên sản phẩm muốn tìm kiếm → Giao diện hiện ra các sản phẩm có tên chứa kí tự mà khách hàng muốn tìm kiếm (cạnh mỗi sản phẩm có nút Thêm giỏ hàng) → Khách hàng click Thêm giỏ hàng → Giao diện thêm giỏ hàng hiện ra với thông tin sản phẩm (ảnh mô tả, tên sản phẩm, tác giả, giá, giá được giảm, mô tả, loại hàng, số lượng), ô nhập số lượng sản phẩm và nút Thêm giỏ hàng → Khách hàng nhập số lượng sản phẩm cần mua, click Thêm giỏ hàng → → Giao diện giỏ hàng hiện ra với các sản phẩm đã được thêm trước đó (có nút checkbox ở bên trái) cùng ô số lượng đã thêm trước đó → Khách hàng tích vào sản phẩm mình cần mua, thay đổi số lượng sản phẩm (nếu cần) → Giao diện hiện

	ra số tiền phải thanh toán và nút Mua → Khách hàng xác nhận và click Mua → Giao diện hiện ra thông tin đơn hàng (sản phẩm, số lượng, số tiền thanh toán) và ô chọn nhà vận chuyển, chọn hình thức thanh toán (tiền mặt hoặc online) và nút Xác nhận → Khách hàng chọn đơn vị vận chuyển, hình thức thanh toán và click nút Xác nhận → Giao diện hiện ra thông tin chi tiết đơn hàng (tên sản phẩm, số lượng sản phẩm, đơn vị vận chuyển, hình thức thanh toán, tổng tiền cần trả) và nút Đặt hàng → Khách hàng xác nhận và click Đặt hàng → Giao diện thông báo đơn hàng được đặt thành công và quay trở về trang chủ của khách hàng.
--	---

Huỷ đơn hàng

Usecase	Huỷ đơn hàng
Actor	Khách hàng
Tiền điều kiện	Tên đăng nhập, mật khẩu
Hậu điều kiện	Huỷ đơn hàng thành công
Hoạt động	khách hàng đăng nhập vào hệ thống → Giao diện trang chủ của khách hàng hiện ra → Khách hàng vào mục đơn hàng của bạn → Giao diện hiện ra thông tin của đơn hàng và trạng thái của đơn hàng , nếu đơn hàng chưa vận chuyển thì có thể click nút Huỷ đơn hàng sáng lên, đơn hàng đã vận chuyển thì nút Huỷ đơn hàng tối lại và không thể click được → Khách hàng click Huỷ đơn hàng → Giao diện hiện ra với thông báo Bạn có chắc chắn huỷ đơn hàng và nút Xác nhận → Khách hàng click nút Xác nhận → Thông báo huỷ đơn hàng thành công và quay trở về trang chủ của khách hàng.

Đăng ký tài khoản/User

Usecase	Đăng ký tài khoản
Actor	Nhân viên
Tiền điều kiện	
Hậu điều kiện	Đăng ký thành công
Hoạt động	Nhân viên chọn chức năng đăng ký tài khoản trên trang chủ của hệ thống → Giao diện đăng ký tài khoản hiện ra với các ô nhập thông tin (tên, số điện thoại, email, mật khẩu) và nút Đăng kí → Nhân viên điền tất cả các thông tin vào các ô sau đó nhấn Đăng ký → Giao diện thông báo đăng ký thành công và quay trở về trang chủ của hệ thống.

Đăng nhập/User

Usecase	Đăng nhập
Actor	Nhân viên
Tiền điều kiện	Tên đăng nhập, mật khẩu
Hậu điều kiện	Đăng nhập thành công
Hoạt động	Nhân viên chọn chức năng đăng nhập tài khoản trên trang chủ của hệ thống → Giao diện đăng nhập hiện ra với các ô nhập thông tin nhân viên (tên, mật khẩu) và nút Đăng nhập và nút Quên mật khẩu → Nhân viên điền thông tin tên đăng nhập, mật khẩu và nhấn nút Đăng nhập → Giao diện hiện thông báo đăng nhập thành công (nếu tên đăng nhập, mật khẩu chính xác) và hiện ra giao diện trang chủ có các sản phẩm hoặc thông báo đăng nhập thất bại (nếu tên đăng nhập, mật khẩu không chính xác) và giao diện đăng nhập hiện ra để người dùng đăng nhập lại.

Usecase	Quên mật khẩu
Actor	Nhân viên
Tiền điều kiện	Tên đăng nhập, mật khẩu
Hậu điều kiện	Tạo mật khẩu thành công
Hoạt động	Nhân viên chọn chức năng đăng nhập tài khoản trên trang chủ của hệ thống → Giao diện đăng nhập hiện ra với các ô nhập thông tin nhân viên (tên, mật khẩu) và nút Đăng nhập và nút Quên mật khẩu → Nhân viên chọn nút Quên mật khẩu → Giao diện lấy lại mật khẩu hiện ra với ô nhập tên và email khôi phục và nút Tiếp theo → Khách hàng nhập thông tin tên đăng nhập và email khôi phục và nhấn nút Tiếp theo → Giao diện hiện ra ô nhập mã xác nhận được gửi về email khôi phục và nút Xác nhận → Nhân viên nhập mã xác nhận và click vào nút Xác nhận → Giao diện hiện ra với ô nhập mật khẩu mới và nút Xác nhận → Nhân viên nhập mật khẩu mới và nút xác nhận → Giao diện hiện ra thông báo thay đổi mật khẩu thành công và quay trở về trang chủ của hệ thống.

Usecase	Thay đổi thông tin cá nhân
Actor	Nhân viên
Tiền điều kiện	Tên đăng nhập, mật khẩu

Hậu điều kiện	Thay đổi thông tin thành công
Hoạt động	Nhân viên đăng nhập vào hệ thống → Giao diện trang chủ của nhân viên hiện ra → Nhân viên chọn thay đổi thông tin cá nhân → Giao diện hiện ra với các ô chứa thông tin nhân viên (tên, số điện thoại, email, mật khẩu) và nút Cập nhật → Nhân viên điền vào các ô mà mình muốn thay đổi và nhấn Cập nhật → Giao diện hiện thông báo Cập nhật thông tin thành công và quay trở về trang chủ của nhân viên.

Usecase	Thêm sản phẩm
Actor	Nhân viên
Tiền điều kiện	Tên đăng nhập, mật khẩu
Hậu điều kiện	Thêm sản phẩm thành công
Hoạt động	Nhân viên đăng nhập vào hệ thống → Giao diện trang chủ của nhân viên hiện ra → Nhân viên click Thêm sản phẩm → Giao diện thêm sản phẩm hiện ra với các ô ảnh mô tả, tên sản phẩm, tác giả, giá, giá được giảm, mô tả, loại hàng, số lượng và nút Thêm sản phẩm → Nhân viên điền các thông tin về sản phẩm vào các ô (không để trống ô nào) và click vào nút Thêm sản phẩm → Thông báo thêm sản phẩm thành công và quay trở lại trang chủ của nhân viên.

Usecase	Sửa sản phẩm
Actor	Nhân viên
Tiền điều kiện	Tên đăng nhập, mật khẩu
Hậu điều kiện	Sửa sản phẩm thành công

Hoạt động	Nhân viên đăng nhập vào hệ thống → Giao diện trang chủ của nhân viên hiện ra cùng ô tìm kiếm và nút Tìm kiếm → Nhân viên nhập tên sản phẩm muốn tìm → Giao diện hiện ra danh sách các sản phẩm có chứa kí tự mà nhân viên nhập gồm ảnh mô tả sản phẩm, tên sản phẩm và nút Cập nhật, nút Xóa, Xem chi tiết → Nhân viên chọn nút Cập nhật → Giao diện hiện ra với các ô về thông tin chi tiết của sản phẩm (ảnh mô tả, tên sản phẩm, tác giả, giá, giá được giảm, mô tả, loại hàng, số lượng) và nút Cập nhật → Nhân viên điền các ô muốn sửa thông tin và nhấn Cập nhật → Giao diện thông báo cập nhật thành công và quay trở về trang chủ của nhân viên.
-----------	---

Usecase	Xoá sản phẩm
Actor	Nhân viên
Tiền điều kiện	Tên đăng nhập, mật khẩu
Hậu điều kiện	Xoá sản phẩm thành công
Hoạt động	Nhân viên đăng nhập vào hệ thống → Giao diện trang chủ của nhân viên hiện ra cùng ô tìm kiếm và nút Tìm kiếm → Nhân viên nhập tên sản phẩm muốn tìm → Giao diện
	hiện ra danh sách các sản phẩm có chứa kí tự mà nhân viên nhập gồm ảnh mô tả sản phẩm, tên sản phẩm và nút Cập nhật, nút Xóa, Xem chi tiết → Nhân viên chọn nút Xóa → Giao diện hiển thị thông báo Bạn có chắc muốn xoá sản phẩm này không? kèm nút Có, nút Không → Nhân viên click Có → Thông báo xoá thành công và quay lại trang chủ của nhân viên.

Usecase	Xem chi tiết sản phẩm
Actor	Nhân viên
Tiền điều kiện	Tên đăng nhập, mật khẩu
Hậu điều kiện	Xem chi tiết sản phẩm thành công

Hoạt động	Nhân viên đăng nhập vào hệ thống → Giao diện trang chủ của nhân viên hiện ra cùng ô tìm kiếm và nút Tìm kiếm → Nhân viên nhập tên sản phẩm muốn tìm → Giao diện hiện ra danh sách các sản phẩm có chứa kí tự mà nhân viên nhập gồm ảnh mô tả sản phẩm, tên sản phẩm và nút Cập nhật, nút Xóa, Xem chi tiết → Nhân viên chọn nút Xem chi tiết → Giao diện hiện ra với các ô về thông tin chi tiết của sản phẩm (ảnh mô tả, tên sản phẩm, tác giả, giá, giá được giảm, mô tả, loại hàng, số lượng) và nút Xóa, nút Cập nhật, nút Thoát → Nhân viên click Thoát → Quay trở về trang chủ của nhân viên.
-----------	---

Usecase	Thêm khách hàng
Actor	Nhân viên
Tiền điều kiện	Tên đăng nhập, mật khẩu
Hậu điều kiện	Thêm khách hàng thành công
Hoạt động	Nhân viên đăng nhập vào hệ thống → Giao diện trang chủ của nhân viên hiện ra → Nhân viên click Thêm khách hàng → Giao diện thêm khách hàng hiện ra với các ô tên, số điện thoại, email, mật khẩu và nút Thêm khách hàng → Nhân viên điền các thông tin về khách hàng vào các ô (không để trống ô nào) và click vào nút Thêm khách hàng → Thông báo thêm khách hàng thành công và quay trở lại trang chủ của nhân viên.

Usecase	Sửa khách hàng
Actor	Nhân viên
Tiền điều kiện	Tên đăng nhập, mật khẩu
Hậu điều kiện	Sửa khách hàng thành công

Hoạt động	Nhân viên đăng nhập vào hệ thống → Giao diện trang chủ của nhân viên hiện ra cùng ô tìm kiếm và nút Tìm kiếm → Nhân viên nhập tên khách hàng muốn tìm → Giao diện hiện ra danh sách các khách hàng có chứa kí tự mà nhân viên nhập gồm tên, số điện thoại, email, mật khẩu và nút Cập nhật, nút Xóa, Xem chi tiết → Nhân viên chọn nút Cập nhật → Giao diện hiện ra với các ô về thông tin chi tiết của khách hàng (tên, số điện thoại, email, mật khẩu) và nút Cập nhật → Nhân viên điền các ô muốn sửa thông tin và nhấn Cập nhật → Giao diện thông báo cập nhật thành công và quay trở về trang chủ của nhân viên.
-----------	---

Usecase	Xoá khách hàng
Actor	Nhân viên
Tiền điều kiện	Tên đăng nhập, mật khẩu
Hậu điều kiện	Xoá khách hàng thành công
Hoạt động	Nhân viên đăng nhập vào hệ thống → Giao diện trang chủ của nhân viên hiện ra cùng ô tìm kiếm và nút Tìm kiếm → Nhân viên nhập tên khách hàng muốn tìm → Giao diện hiện ra danh sách các khách hàng có chứa kí tự mà nhân viên nhập gồm tên, số điện thoại, email, mật khẩu và nút Cập nhật, nút Xóa, Xem chi tiết → Nhân viên chọn nút Xóa → Giao diện hiển thị thông báo Bạn có chắc muốn xóa khách hàng này không? kèm nút Có, nút Không → Nhân viên click Có → Thông báo xóa thành công và quay lại trang chủ của nhân viên.

Usecase	Xem chi tiết khách hàng
Actor	Nhân viên
Tiền điều kiện	Tên đăng nhập, mật khẩu
Hậu điều kiện	Xem chi tiết khách hàng thành công
Hoạt động	Nhân viên đăng nhập vào hệ thống → Giao diện trang chủ của nhân viên hiện ra cùng ô tìm kiếm và nút Tìm kiếm → Nhân viên nhập tên khách hàng muốn tìm → Giao diện hiện ra danh sách các khách hàng có chứa kí tự mà nhân viên nhập gồm tên, số điện thoại, email, mật khẩu và nút Cập nhật, nút Xóa, Xem chi tiết → Nhân viên chọn nút Xem chi tiết → Giao diện hiện ra với các ô về thông tin chi tiết của khách hàng (tên, số điện thoại, email, mật

	khẩu) và nút Xoá, nút Cập nhật, nút Thoát → Nhân viên click Thoát → Quay trở về trang chủ của nhân viên.
--	--

Usecase	Xem chi tiết đơn hàng
Actor	Nhân viên
Tiền điều kiện	Tên đăng nhập, mật khẩu
Hậu điều kiện	Xem đơn hàng thành công
Hoạt động	Nhân viên đăng nhập vào hệ thống → Giao diện trang chủ của nhân viên hiện ra → Nhân viên click Đơn hàng → Giao diện đơn hàng hiện ra với danh sách các đơn hàng và ô tìm kiếm (theo tên khách hàng hoặc mã đơn hàng), nút Tìm kiếm → Nhân viên nhập tên khách hàng hoặc mã đơn hàng và click nút Tìm kiếm → Giao diện hiện ra với danh sách các đơn hàng và nút Xem chi tiết → Nhân viên click Xem chi tiết vào đơn hàng mình muốn xem → Giao diện đơn hàng hiện ra với thông tin mã đơn hàng, tên khách hàng, tên sản phẩm, số lượng, giá mỗi sản phẩm, tổng tiền thanh toán, đơn vị vận chuyển, hình thức thanh toán và nút Thoát → Nhân viên click Thoát → Giao diện quay về trang chủ của nhân viên.

Usecase	Xác nhận đơn hàng
Actor	Nhân viên
Tiền điều kiện	Tên đăng nhập, mật khẩu
Hậu điều kiện	Xác nhận đơn hàng thành công
Hoạt động	Nhân viên đăng nhập vào hệ thống → Giao diện của nhân viên hiện ra → Nhân viên chọn xác nhận đặt hàng → Giao diện hiện ra danh sách các đơn hàng đã đặt của khách hàng và nút Xác nhận, Huỷ bỏ → Nhân viên click nút xác nhận cho từng đơn hàng và cập nhật dữ liệu để khách hàng dễ dàng theo dõi vận chuyển đơn hàng của mình → Thông báo thành công và quay về trang chủ các đơn hàng cần xác nhận.

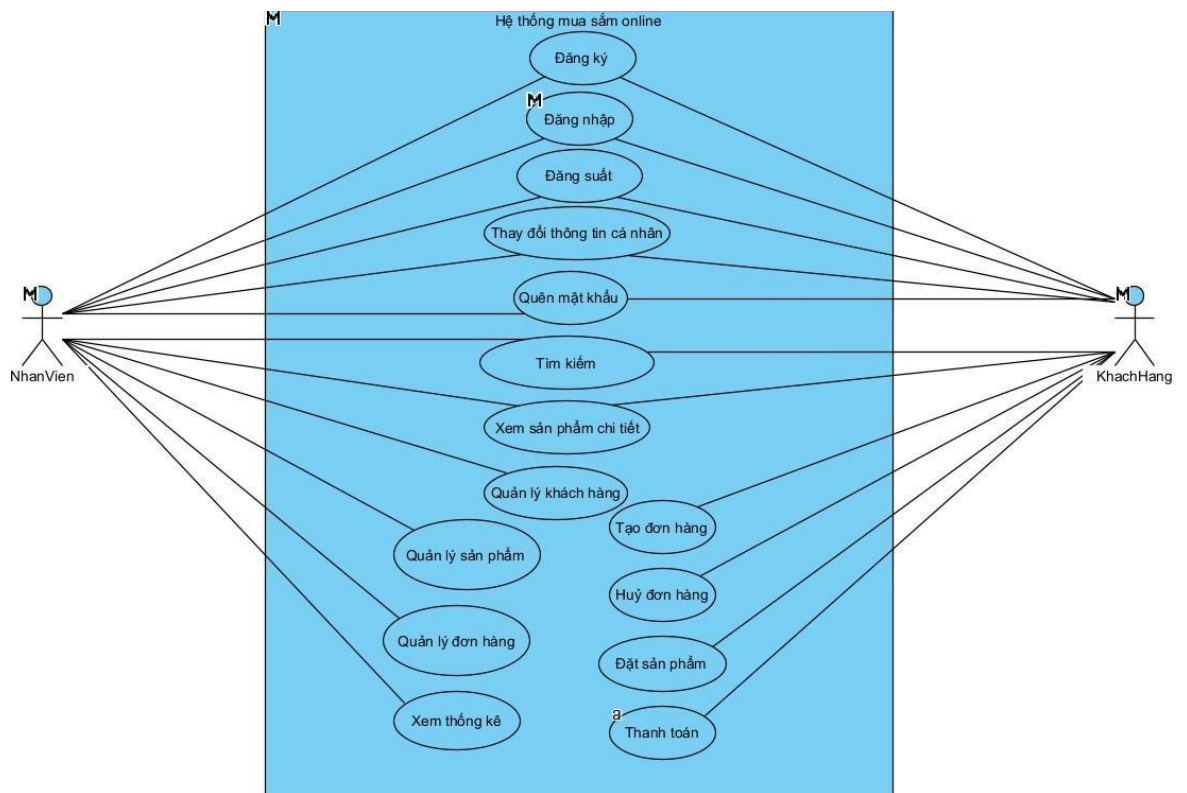
Usecase	Huỷ bỏ đơn hàng
Actor	Nhân viên

Tiền điều kiện	Tên đăng nhập, mật khẩu
Hậu điều kiện	Huỷ đơn hàng thành công
Hoạt động	Nhân viên đăng nhập vào hệ thống → Giao diện của nhân viên hiện ra → Nhân viên chọn xác nhận đơn hàng → Giao diện hiện ra danh sách các đơn hàng đã đặt của khách hàng và nút Xác nhận, Huỷ bỏ → Nhân viên click nút Huỷ bỏ cho đơn hàng muốn huỷ và cập nhật dữ liệu để khách hàng dễ dàng theo dõi vận chuyển đơn hàng của mình → Thông báo thành công và quay về trang chủ các đơn hàng cần xác nhận.

Usecase	Thống kê doanh thu
Actor	Nhân viên
Tiền điều kiện	Tên đăng nhập, mật khẩu
Hậu điều kiện	Xem thống kê thành công
Hoạt động	Nhân viên đăng nhập vào hệ thống → Giao diện của nhân viên hiện ra → Nhân viên chọn Xem thống kê doanh thu → Giao diện hiện ra ô nhập từ ngày, đến ngày và nút Xem thống kê → Nhân viên nhập ngày bắt đầu, ngày kết thúc và click Xem thống kê → Giao diện hiện ra tổng doanh thu và danh sách các đơn hàng trong thời gian nhân viên muốn tìm kiếm.

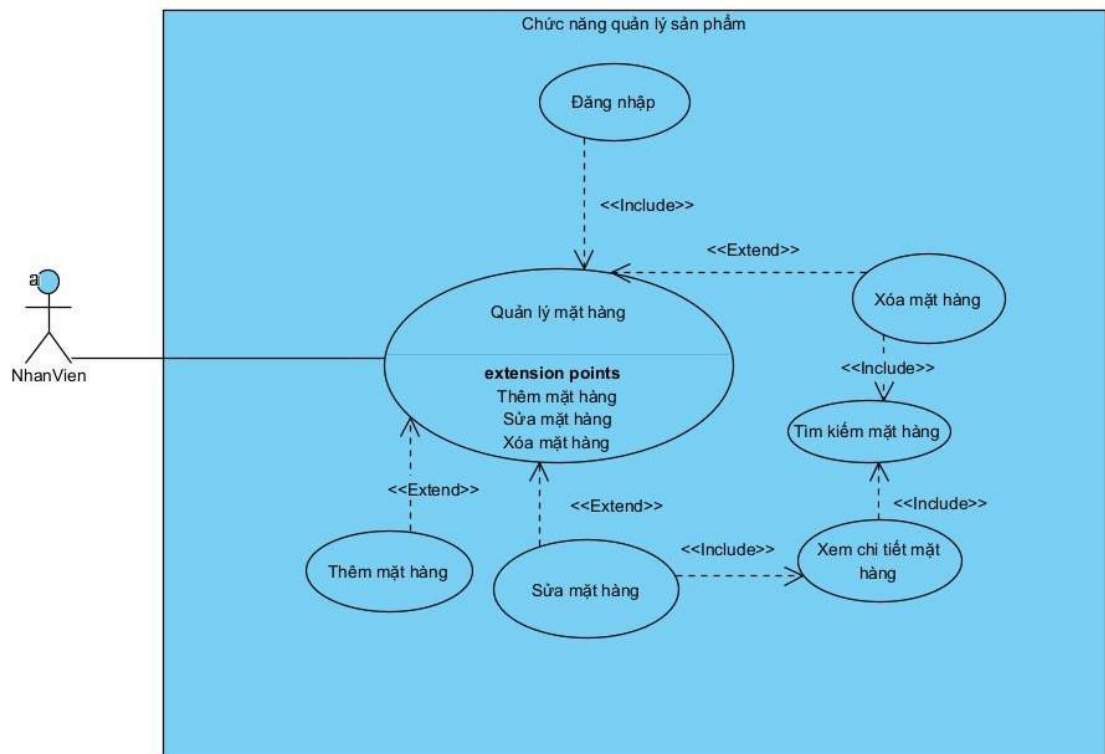
Câu 2: Vẽ Biểu đồ use case tổng quát và biểu đồ usecase chi tiết cho từng chức năng/dịch vụ

- Biểu đồ use case tổng quát:

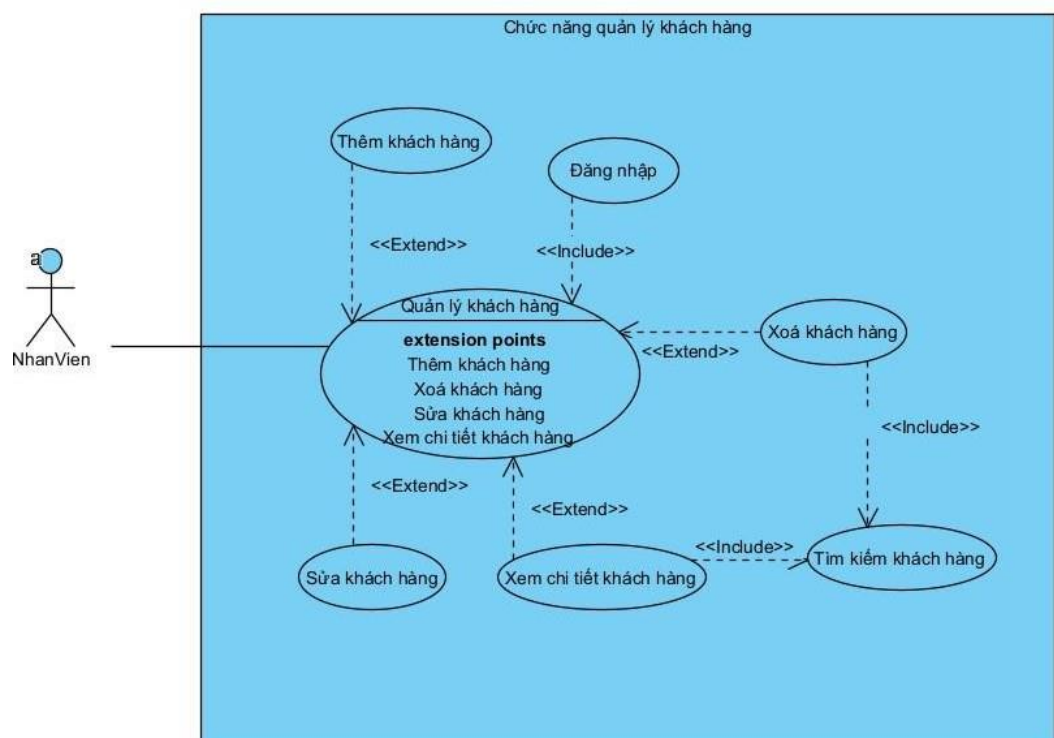


- Biểu đồ use case cho từng chức năng:

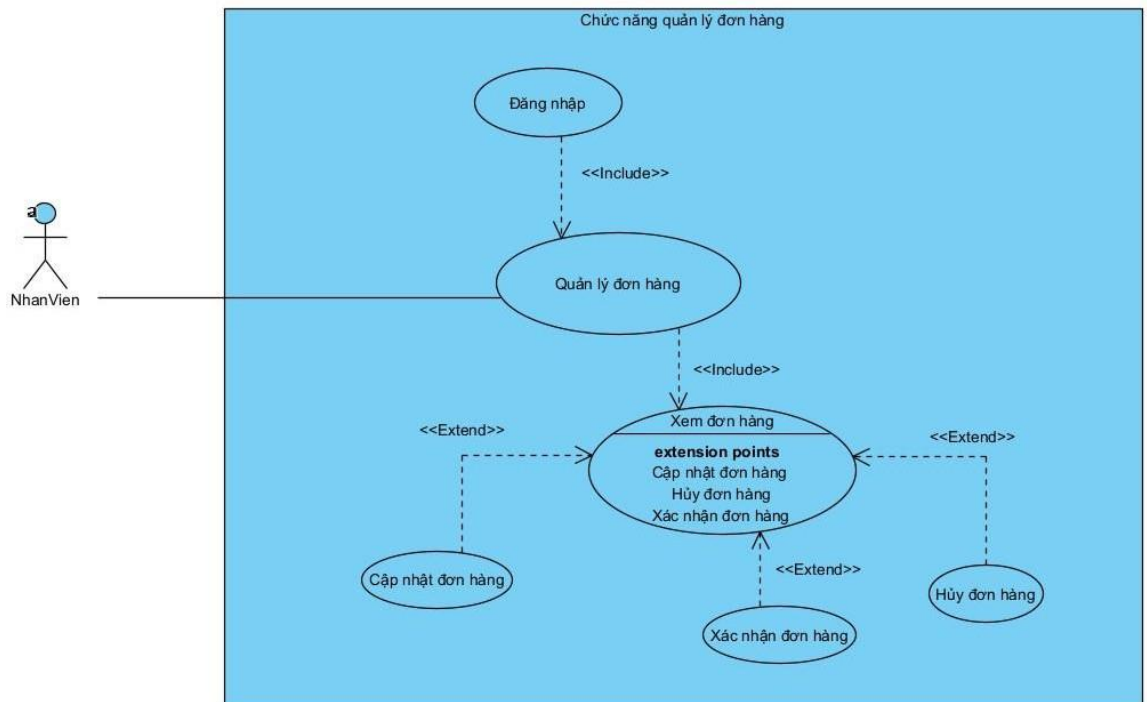
+ Chức năng quản lý sản phẩm



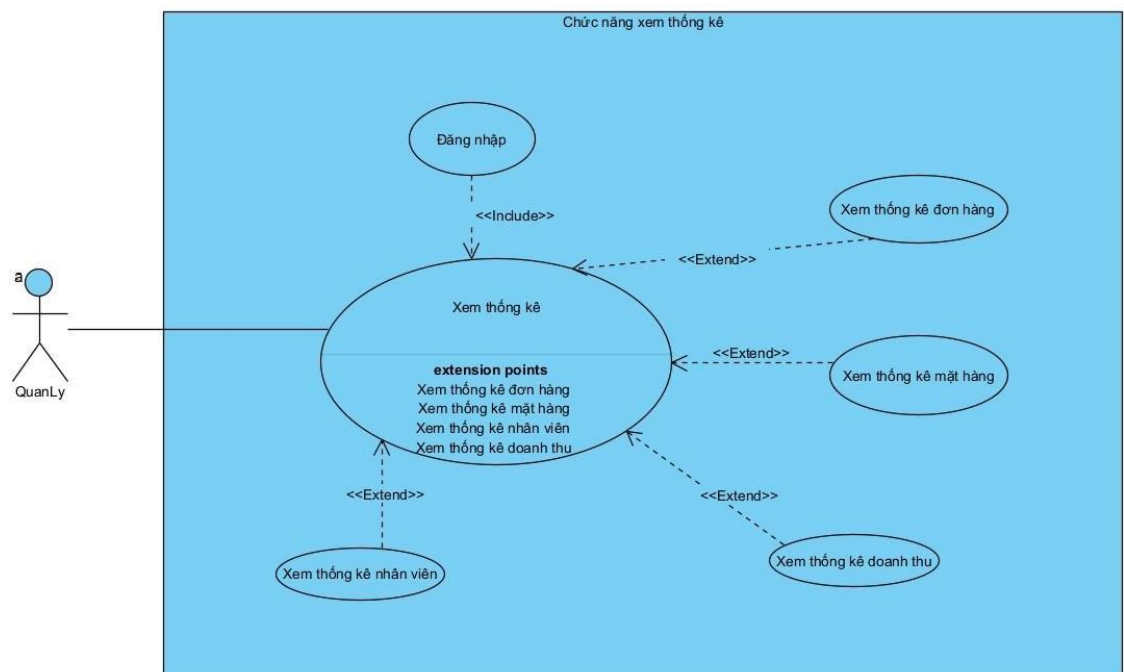
+ Chức năng quản lý khách hàng



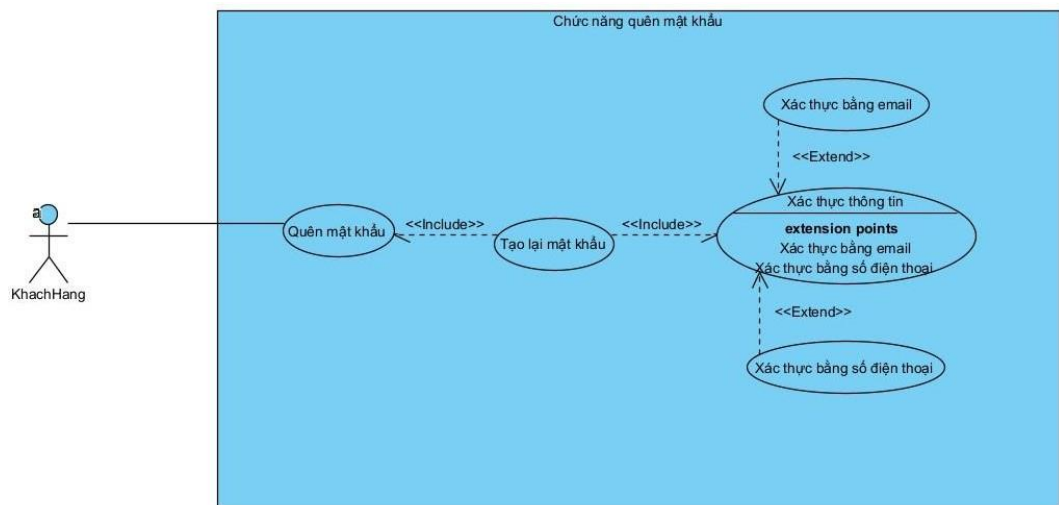
+ Chức năng quản lý đơn hàng



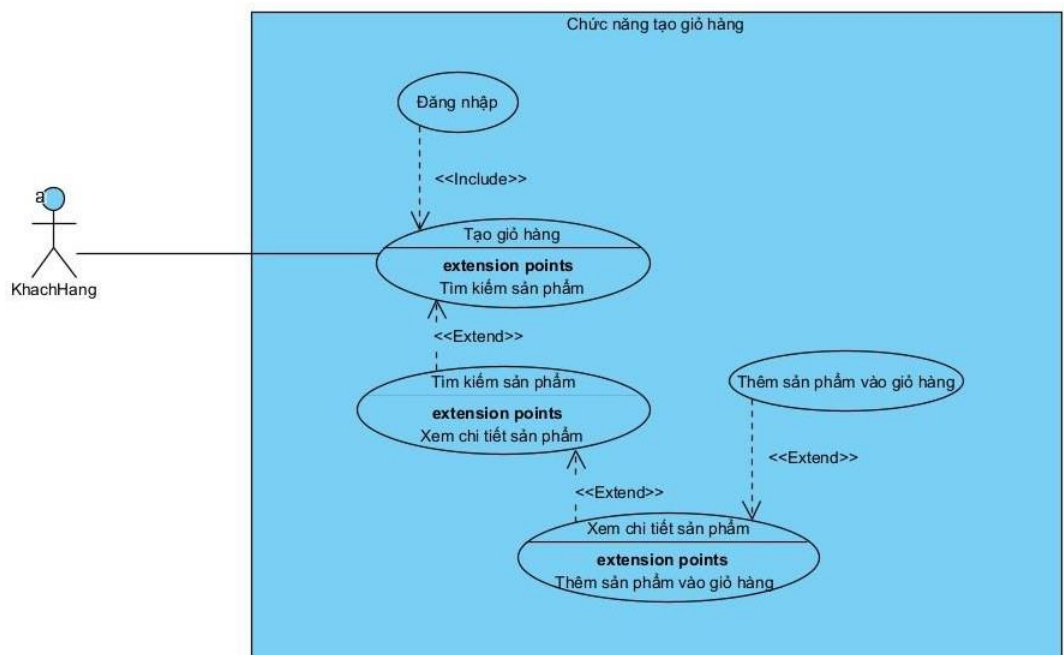
+ Chức năng xem thống kê



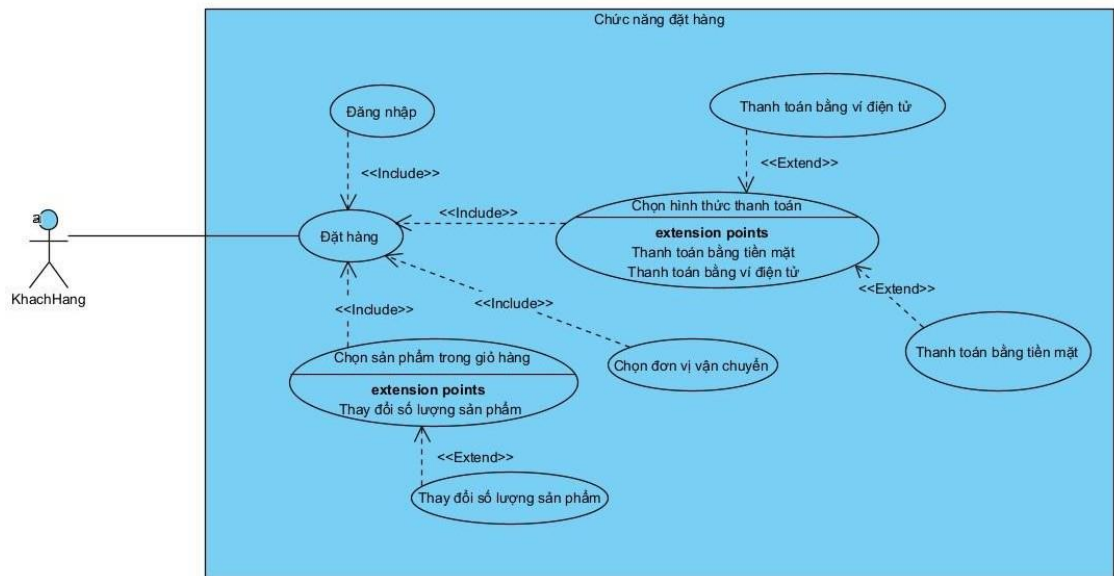
+ Chức năng quên mật khẩu



+ Chức năng tạo giỏ hàng



+ Chức năng đặt hàng



Câu 3: Vẽ biểu đồ phân rã Hệ ecomSys thành các service và các tương tác giữa các dịch vụ (sử dụng quan hệ <> trong UML)

Ta sẽ phân rã dịch vụ như sau:

+ Dịch vụ người dùng: Dịch vụ này liên quan đến việc quản lý tài khoản người dùng và xác thực, chẳng hạn như cho phép người dùng tạo và đăng nhập vào tài khoản của họ cũng như lưu trữ thông tin hồ sơ người dùng.

+ Dịch vụ sản phẩm: Dịch vụ này liên quan đến việc quản lý sản phẩm có sẵn để bán trên trang web, chẳng hạn như thêm sản phẩm mới, cập nhật sản phẩm, xóa sản phẩm hay xem chi tiết sản phẩm.

+ Dịch vụ tìm kiếm: Dịch vụ này liên quan đến việc tìm kiếm các sản phẩm trên trang web để xem chi tiết, thêm giỏ hàng hoặc mua sắm. Có thể tìm kiếm bằng text.

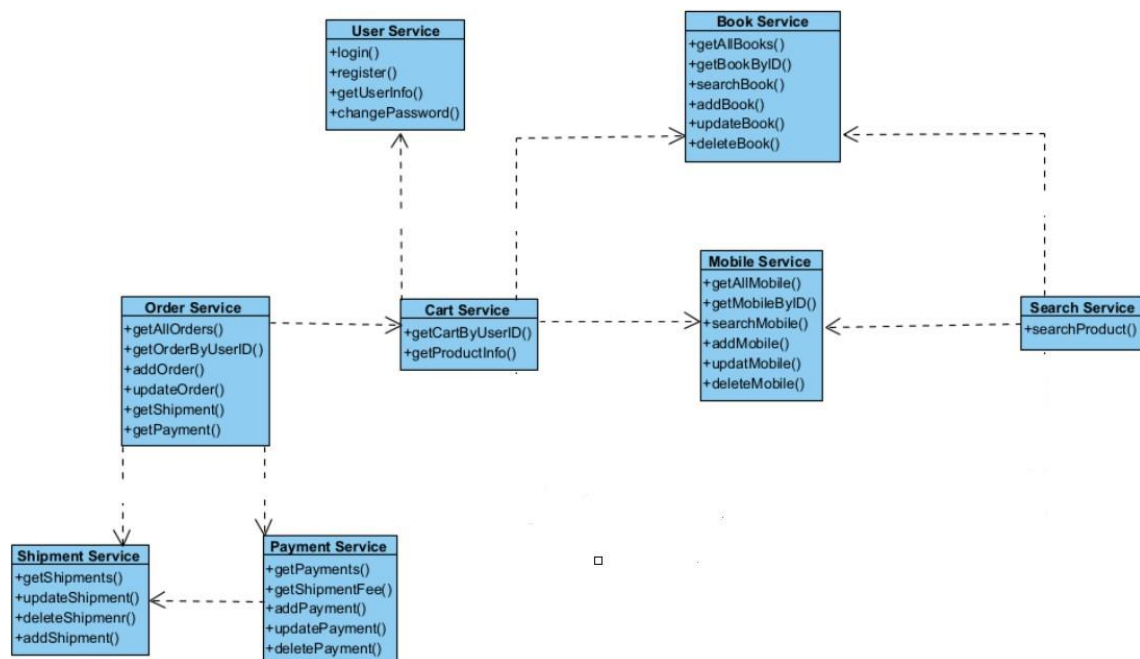
+ Dịch vụ giỏ hàng: Dịch vụ này có khả năng chịu trách nhiệm quản lý các chức năng giỏ hàng của trang web, chẳng hạn như cho phép người dùng thêm các mặt hàng vào giỏ hàng của họ, cập nhật giỏ hàng khi các mặt hàng được thêm vào hoặc xóa và tính toán tổng chi phí của đơn hàng.

+ Dịch vụ đặt hàng: Dịch vụ này liên quan đến quản lý quy trình đặt hàng và thực hiện đơn hàng, chẳng hạn như nhận và xử lý đơn hàng, theo dõi mức tồn kho và cập nhật cho khách hàng về trạng thái đơn hàng.

+ Dịch vụ thanh toán: Dịch vụ này sẽ xử lý quá trình thanh toán, chẳng hạn như chấp nhận thanh toán bằng tài khoản online (ví điện tử), xác minh tính hợp lệ của thông tin thanh toán và bắt đầu giao dịch với cổng thanh toán hoặc ngân hàng.

+ Dịch vụ vận chuyển: Dịch vụ này sẽ quản lý việc vận chuyển và giao sản phẩm cho khách hàng, chẳng hạn như theo dõi các gói hàng, phối hợp với hãng vận chuyển và tính toán chi phí vận chuyển.

+ Dịch vụ nhân viên: Dịch vụ này có thể liên quan đến việc quản lý tài khoản và quyền của nhân viên, chẳng hạn như cho phép nhân viên truy cập vào một số phần nhất định của trang web và theo dõi chỉ số hiệu suất của nhân viên.



Câu 4: Trình bày các dạng communication giữa các service với nhau (synchronous và asynchronous) với code và ví dụ.

Trong môi trường phân tán và dựa trên dịch vụ, các dịch vụ thường cần giao tiếp với nhau để hoạt động hiệu quả. Có hai dạng chính của giao tiếp giữa các dịch vụ: đồng bộ (synchronous) và không đồng bộ (asynchronous). Dưới đây là một số ví dụ và mã nguồn minh họa cho cả hai dạng giao tiếp này.

Giao tiếp Đồng bộ (Synchronous Communication)

- HTTP/HTTPS Request-Response: Trong giao thức HTTP, dịch vụ gửi một yêu cầu (request) đến một dịch vụ khác và đợi cho đến khi nhận được một phản hồi (response) trước khi tiếp tục thực hiện. Quá trình này là đồng bộ vì dịch vụ gửi yêu cầu phải chờ cho đến khi nhận được phản hồi.

- Remote Procedure Call (RPC): RPC là một phương thức giao tiếp mà một dịch vụ gửi một yêu cầu cho một dịch vụ khác để thực thi một hàm hoặc phương thức cụ thể và chờ đợi kết quả trước khi tiếp tục thực hiện. Quá trình này cũng là đồng bộ vì dịch vụ gọi trực tiếp chờ đợi cho đến khi hàm hoặc phương thức được thực thi và kết quả được trả về.
- WebSocket Communication: Trong môi trường WebSocket, dịch vụ có thể thiết lập một kết nối hai chiều và gửi và nhận dữ liệu một cách liên tục. Mặc dù WebSocket cho phép giao tiếp real-time, nhưng quá trình gửi và nhận dữ liệu vẫn là đồng bộ.
- Giao tiếp đồng bộ là khi các dịch vụ chờ đợi lẫn nhau để trao đổi dữ liệu hoặc hoàn thành một tác vụ trước khi tiếp tục công việc tiếp theo. Dưới đây là một ví dụ về giao tiếp đồng bộ bằng cách sử dụng REST API và gọi HTTP request:

```
import requests

# URL của REST API endpoint
url = 'https://api.example.com/data'

# Gửi một GET request để lấy dữ liệu từ API (đồng bộ)
response = requests.get(url)

# Xử lý dữ liệu trả về từ API
if response.status_code == 200:
    data = response.json()
    print(data)
else:
    print('Failed to fetch data:', response.status_code)
```

Trong ví dụ trên, chương trình chờ đợi cho đến khi nhận được phản hồi từ API trước khi tiếp tục thực hiện các lệnh tiếp theo.

Không đồng bộ (Asynchronous) Communication:

- Message Queues: Trong hệ thống hàng đợi tin nhắn, dịch vụ gửi tin nhắn đến một hàng đợi mà không cần chờ đợi cho đến khi tin nhắn được xử lý. Dịch vụ khác có thể nhận và xử lý các tin nhắn từ hàng đợi một cách không đồng bộ, có nghĩa là chúng có thể tiếp tục thực hiện các công việc khác trong khi đang chờ đợi tin nhắn.
- Event Streams: Trong một môi trường dữ liệu dòng sự kiện, dịch vụ có thể gửi và nhận các sự kiện một cách không đồng bộ. Các sự kiện có thể được

gửi và xử lý ngay khi chúng xảy ra mà không cần chờ đợi cho đến khi dịch vụ khác phản hồi.

- **Callback Functions:** Trong một số trường hợp, các dịch vụ có thể sử dụng callback functions để thực hiện giao tiếp không đồng bộ. Dịch vụ gửi yêu cầu và chỉ định một hàm callback để xử lý kết quả khi nó được trả về.
- **Giao tiếp không đồng bộ** là khi các dịch vụ không chờ đợi nhau mà có thể tiếp tục thực hiện công việc khác trong khi đang chờ dữ liệu hoặc phản hồi từ dịch vụ khác. Dưới đây là một ví dụ về giao tiếp không đồng bộ bằng cách sử dụng Python và thư viện `asyncio` để tạo ra hai coroutine chạy không đồng bộ:

```
import asyncio

# Coroutine để gửi request và xử lý dữ liệu
async def fetch_data(url):
    print('Fetching data from:', url)
    # Giả định có một hàm async để gửi request và nhận dữ liệu
    data = await send_request(url)
    print('Received data:', data)

# Coroutine để thực hiện công việc khác trong khi chờ
async def do_other_work():
    while True:
        print('Doing other work...')
        await asyncio.sleep(1) # Giả định công việc khác

# Tạo một event loop và chạy các coroutine
async def main():
    url = 'https://api.example.com/data'
    task1 = asyncio.create_task(fetch_data(url))
    task2 = asyncio.create_task(do_other_work())
    await asyncio.gather(task1, task2)

# Chạy event loop để thực thi main coroutine
asyncio.run(main())
```

Trong ví dụ này, coroutine **`fetch_data`** gửi một request nhưng không chờ đợi phản hồi và tiếp tục thực hiện các công việc khác trong khi chờ dữ liệu từ API.

Câu 5: Trình bày sử dụng các dạng communication giữa các service với code cho hệ ecomSys

+ Giao tiếp giữa manager và book trong xóa sách

```
class DeleteCategory(APIView):
    def delete(self, request, category_id):
        token_verification_url = "http://localhost:4001/api/ecomSys/manager/verify-token/"
        headers = {'Authorization': request.headers.get('Authorization')}
        response = requests.get(token_verification_url, headers=headers)

        if response.status_code == 200:
            try:
                category = Category.objects.get(category_id=category_id)
            except Category.DoesNotExist:
                return Response({'error': 'Category not found'}, status=status.HTTP_404_NOT_FOUND)

            serializer = CategorySerializer()
            serializer.destroy(category)

            return Response({'message': 'Category soft deleted'}, status=status.HTTP_204_NO_CONTENT)

        return Response({'error': 'Invalid token.'}, status=status.HTTP_401_UNAUTHORIZED)
```

+ Giao tiếp giữa manager và type sách

```
class DeleteCategory(APIView):
    def delete(self, request, category_id):
        token_verification_url = "http://localhost:4001/api/ecomSys/manager/verify-token/"
        headers = {'Authorization': request.headers.get('Authorization')}
        response = requests.get(token_verification_url, headers=headers)

        if response.status_code == 200:
            try:
                category = Category.objects.get(category_id=category_id)
            except Category.DoesNotExist:
                return Response({'error': 'Category not found'}, status=status.HTTP_404_NOT_FOUND)

            serializer = CategorySerializer()
            serializer.destroy(category)

            return Response({'message': 'Category soft deleted'}, status=status.HTTP_204_NO_CONTENT)

        return Response({'error': 'Invalid token.'}, status=status.HTTP_401_UNAUTHORIZED)
```

+ Giao tiếp giữa type , addBook và manager


```

class CreateCategoryView(APIView):
    def post(self, request):
        token_verification_url = "http://localhost:4001/api/ecomSys/manager/verify-token/"
        headers = {'Authorization': request.headers.get('Authorization')}
        response = requests.get(token_verification_url, headers=headers)

        if response.status_code == 200:
            serializer = CategorySerializer(data=request.data)
            if serializer.is_valid():
                serializer.save()
                return Response(serializer.data, status=status.HTTP_201_CREATED)
            return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
        return Response({'error': 'Invalid token.'}, status=status.HTTP_401_UNAUTHORIZED)

class AddBookView(APIView):
    def post(self, request):
        token_verification_url = "http://localhost:4001/api/ecomSys/manager/verify-token/"
        headers = {'Authorization': request.headers.get('Authorization')}
        response = requests.get(token_verification_url, headers=headers)

        if response.status_code == 200:
            serializer = BookSerializer(data=request.data, context={'request': request})
            if serializer.is_valid():
                serializer.save()
                return Response(serializer.data, status=status.HTTP_201_CREATED)
            return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
        return Response({'error': 'Invalid token.'}, status=status.HTTP_401_UNAUTHORIZED)

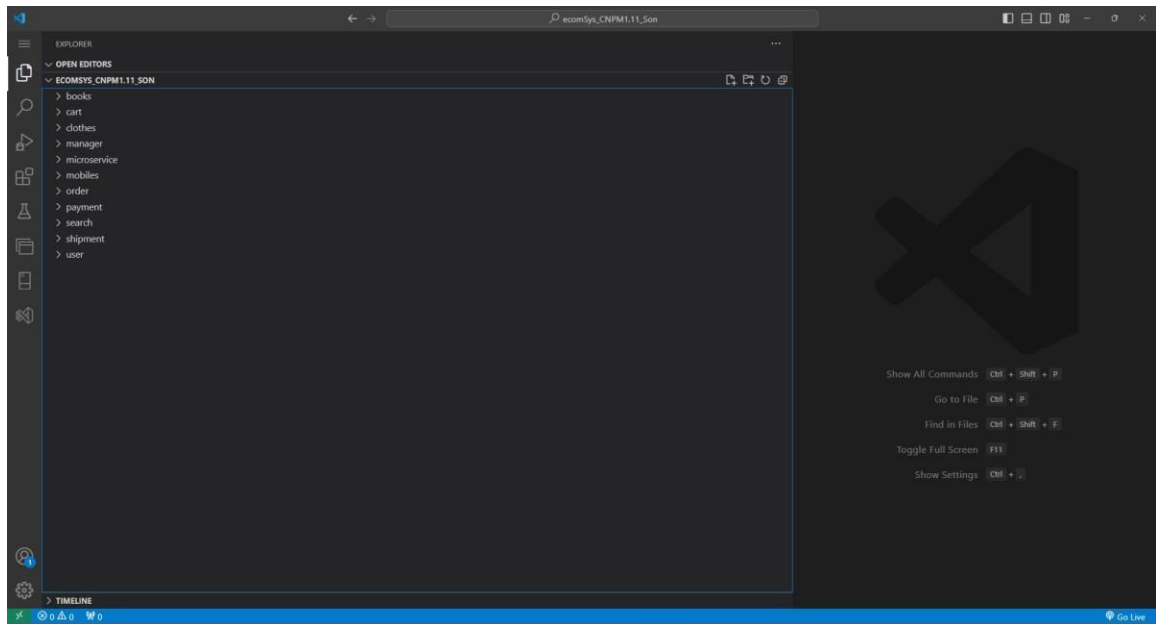
```

Bài tập 4

I. Tạo 10 service

- Tạo 1 project với tên `ecomSys_CNPM5.07_quy`
- Mở Command Prompt ở máy và chuyển đến project `ecomSys_CNPM2.03_nhi` bằng lệnh `cd "path/ecomSys_CNPM5.07_quy"`
- Tạo môi trường ảo "microservice" để làm việc với project bằng lệnh `python -m venv microservice`
- Chạy môi trường ảo với lệnh `microservice\Scripts\activate.bat`
- Sau khi chạy môi trường ảo, dùng lệnh `pip` để cài đặt các thư viện python:
 - o Cài django: `pip install django==4.0.1` (Tương thích với django 1.3.6)
 - o Cài django: `pip install django`
 - o Cài pymongo: `pip install pymongo==3.12.3` (Tương thích với django 4.0.1)
 - o Cài MySQL Client: `pip install mysqlclient`
 - o Cài rest_framework: `pip install djangorestframework`
 - o Cài corsheaders: `pip install django-cors-headers`
 - o Cài simplejwt: `pip install djangorestframework-simplejwt`
- Sau khi đã cài các thư viện cần thiết, dùng lệnh `django-admin startproject` để tạo các service
 - o Tạo user service: `django-admin startproject user`
 - o Tạo manager service: `django-admin startproject manager`
 - o Tạo books service: `django-admin startproject books`
 - o Tạo clothers

service: *django-admin startproject clothers* ○ Tạo mobiles
 service: *django-admin startproject mobiles* ○ Tạo search
 service: *django-admin startproject search* ○ Tạo cart service:
django-admin startproject cart ○ Tạo order service: *django-*
admin startproject order ○ Tạo payment service: *django-admin*
startproject payment ○ Tạo shipment service: *django-admin*
startproject shipment - Sau khi tạo xong sẽ có cấu trúc project
 như sau:

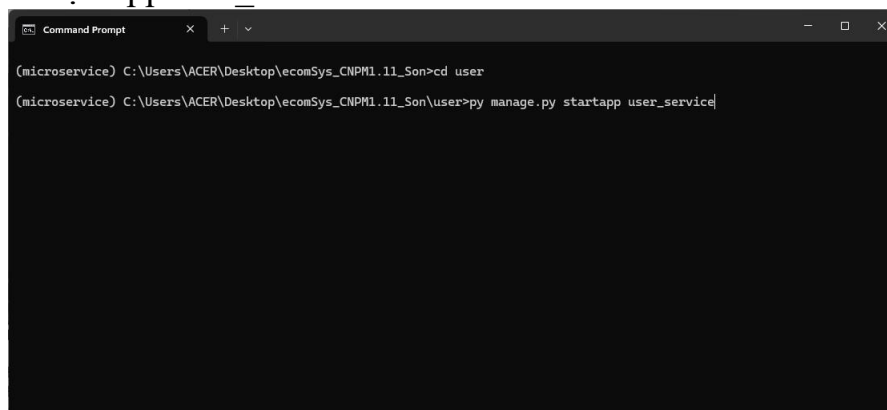


II. Tạo Rest API cho các service

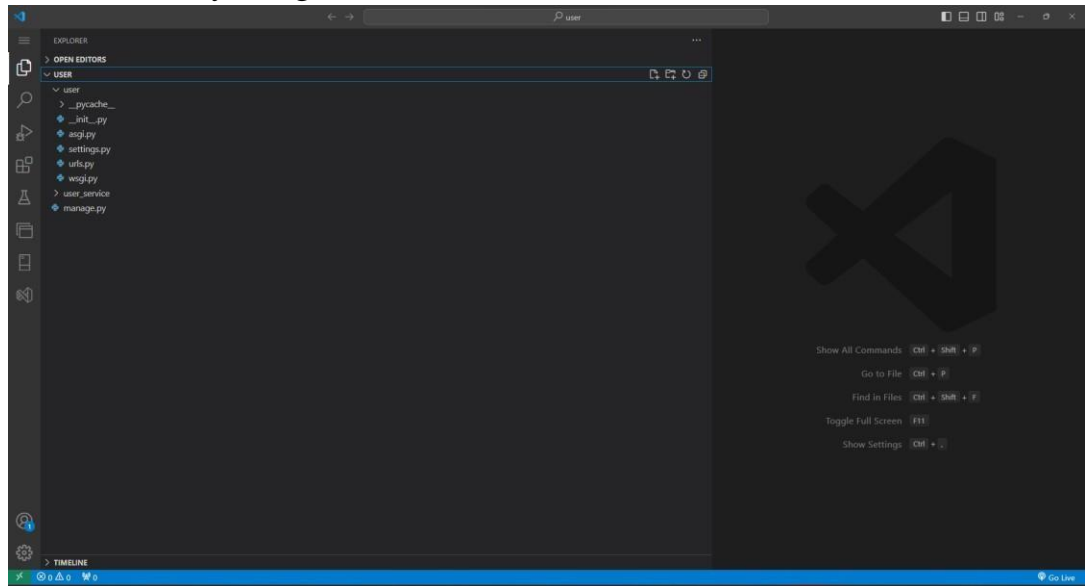
Trong project này có một số tệp cần được chú trọng. Thứ nhất là *models.py* được sử dụng để định nghĩa mô hình dữ liệu và quan hệ của chúng. Thứ hai là *serializers.py* được sử dụng để chuyển đổi dữ liệu *python* thành dữ liệu *json* (trong trường hợp này), cũng như kiểm tra tính hợp lệ của dữ liệu đầu vào. Thứ ba là *views.py* được sử dụng để định nghĩa và xử lý yêu cầu đầu vào đồng thời tương tác với mô hình dữ liệu để tạo và gửi phản hồi phù hợp.

1. User Service

- Chuyển hướng đến *user service* từ màn hình Command Prompt bằng lệnh *cd user*. Sau đó chạy lệnh *py manage.py startapp user_service* để tạo app *user_service*



- Sau khi chạy xong, cấu trúc thư mục user service như sau:



- Vào *settings.py* để thay đổi một số cấu hình: ○ Thêm cổng PORT = 4000

```
ALLOWED_HOSTS = ['localhost']  
PORT = 4000
```

○ Sửa SECRET_KEY và thêm REFRESH_TOKEN_SECRET để phục vụ việc mã hóa dữ liệu user

```
# SECURITY WARNING: keep the secret key used in production secret!  
SECRET_KEY = '@Helios'  
REFRESH_TOKEN_SECRET = 'hong'
```

○ Thêm *corsheaders*, *rest_framework*, *user_service* vào
INSTALLED_APPS

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'corsheaders',  
    'rest_framework',  
    'user_service',  
]
```

○ Cấu hình cơ sở dữ liệu phù hợp với MySQL

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'db_user',
        'USER': 'userservice',
        'PASSWORD': '12345',
        'HOST': 'localhost',
        'PORT': '3306'
    }
}
```

- Thêm `CORS_ALLOW_CREDENTIALS` và `CORS_ORIGIN_WHITELIST` để quản lý cấu hình Cross-Origin Resource Sharing (CORS), một cơ chế cho phép trình duyệt web yêu cầu tài nguyên từ một domain khác với nguồn mà trang web đang chạy

```
CORS_ALLOW_CREDENTIALS = True # to accept cookies via ajax request
CORS_ORIGIN_WHITELIST = [
    'http://localhost:3000' # the domain for front-end app(you can add more than 1)
]
```

- Vào `models.py` của app `user_service` để tạo model `User` kế thừa từ `AbstractUser`

```
class User(AbstractUser):
    phone_number = models.CharField(max_length=20)
    email = models.EmailField(max_length=255, unique=True)
    username = models.CharField(max_length=100, unique=False, null=True, default=None)

    USERNAME_FIELD = 'email'
    REQUIRED_FIELDS = ['username']

    def __str__(self):
        return self.email
```

- Vào lại `settings.py` thêm thông tin cấu hình để xác định mô hình `User` tùy chỉnh mà Django sẽ dùng để xác thực
- Quay lại màn hình Command Prompt chạy các lệnh sau:
 - `py manage.py makemigration`
 - `py manage.py migrate`
- Tạo `serializers.py` ở app `user_service` rồi thêm vào các thay đổi sau:
 - Tạo `UserSerializer` để xác định cách nhận vào, lưu và trả về dữ liệu `User` khi đăng ký


```
class UserSerializer(serializers.ModelSerializer):
    password = serializers.CharField(write_only=True)

    class Meta:
        model = User
        fields = ['id', 'first_name', 'last_name', 'phone_number', 'email', 'password']

    def create(self, validated_data):
        validated_data['password'] = make_password(validated_data.get('password'))
        return super().create(validated_data)
```

- Tạo UserLoginSerializer để xác định cách nhận vào và trả về dữ liệu User khi đăng nhập

```
class UserLoginSerializer(serializers.Serializer):
    email = serializers.CharField()
    password = serializers.CharField(write_only=True)

    def validate(self, data):
        email = data.get('email', None)
        password = data.get('password', None)

        if not email or not password:
            raise serializers.ValidationError('Email and password are required.')

        user = User.objects.filter(email=email).first()
        if user:
            if check_password(password, user.password):
                return user
            else:
                raise serializers.ValidationError('Invalid password.')
        else:
            raise serializers.ValidationError('User does not exist.')
```

- Tạo UserInfoSerializer để xác định cách trả về dữ liệu User khi xem thông tin

```
class UserInfoSerializer(serializers.ModelSerializer):
    class Meta:
        model = User
        fields = ['id', 'first_name', 'last_name', 'phone_number', 'email']
```

- Tạo ChangePasswordSerializer để xác định cách nhận vào và trả về dữ liệu User khi đổi mật khẩu

```
class ChangePasswordSerializer(serializers.Serializer):
    old_password = serializers.CharField()
    new_password = serializers.CharField()

    def validate_old_password(self, value):
        user = self.context['request'].user
        if not check_password(value, user.password):
            raise serializers.ValidationError('Incorrect old password.')
        return value
```

- Tạo UpdateProfileSerializer để xác định cách nhận vào và trả về dữ liệu User khi thay đổi thông tin

```
class UpdateProfileSerializer(serializers.ModelSerializer):
    class Meta:
        model = User
        fields = ['first_name', 'last_name', 'phone_number']

    def update(self, instance, validated_data):
        instance.first_name = validated_data.get('first_name', instance.first_name)
        instance.last_name = validated_data.get('last_name', instance.last_name)
        instance.phone_number = validated_data.get('phone_number', instance.phone_number)
        instance.save()
        return instance
```

- Tạo *utils.py* ở app *user_service* để xác định cách mã hóa
 - o Tạo *access_token* dựa trên thông tin người dùng và thời gian hết hạn

```
def generate_access_token(user):
    access_token_payload = {
        'user_id': user.id,
        'exp': datetime.datetime.utcnow() + datetime.timedelta(days=1),
        'iat': datetime.datetime.utcnow(),
    }
    access_token = jwt.encode(access_token_payload,
                              settings.SECRET_KEY, algorithm='HS256').decode('utf-8')
    return access_token
```

- o Tạo *refresh_token* dựa trên thông tin người dùng và thời gian hết hạn

```
def generate_refresh_token(user):
    refresh_token_payload = {
        'user_id': user.id,
        'exp': datetime.datetime.utcnow() + datetime.timedelta(days=7),
        'iat': datetime.datetime.utcnow()
    }
    refresh_token = jwt.encode(
        refresh_token_payload, settings.REFRESH_TOKEN_SECRET, algorithm='HS256').decode('utf-8')
    return refresh_token
```

- Tạo *authentication.py* ở app *user_service* để tạo xác thực tùy chỉnh

```

class SafeJWTAuthentication(BaseAuthentication):
    def authenticate(self, request):
        User = get_user_model()
        authorization_header = request.headers.get('Authorization')

        if not authorization_header:
            return None

        try:
            token_prefix, access_token = authorization_header.split(' ')
            if token_prefix != 'Bearer':
                raise exceptions.AuthenticationFailed('Invalid token prefix')

            payload = jwt.decode(
                access_token, settings.SECRET_KEY, algorithms=['HS256'])
        except jwt.ExpiredSignatureError:
            raise exceptions.AuthenticationFailed('Access token expired')
        except ValueError:
            raise exceptions.AuthenticationFailed('Invalid token format')
        except jwt.InvalidTokenError:
            raise exceptions.AuthenticationFailed('Invalid token')

        user_id = payload.get('user_id')
        if user_id is None:
            raise exceptions.AuthenticationFailed('User ID not found in token payload')

        user = User.objects.filter(id=user_id).first()
        if user is None:
            raise exceptions.AuthenticationFailed('User not found')

        if not user.is_active:
            raise exceptions.AuthenticationFailed('User is inactive')

        return (user, None)

```

- Vào settings.py thêm cấu hình để phục vụ xác thực. Ở đây tôi chọn xác thực tùy chỉnh

```

REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': (
        'user_service.authentication.SafeJWTAuthentication',
    ),
    'DEFAULT_PERMISSION_CLASSES': (
        'rest_framework.permissions.IsAuthenticated',
    )
}

```

- Vào views.py của app *user_service* để thực hiện các logic xử lý dữ liệu từ yêu cầu và xác định cách phản hồi
 - o Chức năng đăng ký

```
class RegisterView(APIView):
    permission_classes = [AllowAny]
    def post(self, request):
        serializer = UserSerializer(data=request.data)
        if serializer.is_valid():
            user = serializer.save()
            access_token = generate_access_token(user)
            refresh_token = generate_refresh_token(user)

            user_serializer = UserInfoSerializer(user)

            return Response({
                'user': user_serializer.data,
                'refresh': refresh_token,
                'access': access_token,
            }, status=status.HTTP_201_CREATED)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

○ Chức năng đăng nhập

```
class LoginView(APIView):
    permission_classes = [AllowAny]
    def post(self, request):
        serializer = UserLoginSerializer(data=request.data)
        if serializer.is_valid():
            user = serializer.validated_data
            access_token = generate_access_token(user)
            refresh_token = generate_refresh_token(user)

            user_serializer = UserInfoSerializer(user)

            return Response({
                'user': user_serializer.data,
                'refresh': refresh_token,
                'access': access_token,
            }, status=status.HTTP_200_OK)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

○ Chức năng xem thông tin (Cần đăng nhập và xác thực)

```
class UserInfoView(APIView):
    permission_classes = [IsAuthenticated]
    authentication_classes = [SafeJWTAuthentication]

    def get(self, request):
        user = request.user
        serializer = UserInfoSerializer(user)
        return Response(serializer.data, status=status.HTTP_200_OK)
```

○ Chức năng đổi mật khẩu (Cần đăng nhập và xác thực)

```
class ChangePasswordView(APIView):
    permission_classes = [IsAuthenticated]
    authentication_classes = [SafeJWTAuthentication]

    def post(self, request):
        serializer = ChangePasswordSerializer(data=request.data, context={'request': request})
        if serializer.is_valid():
            user = request.user
            user.set_password(serializer.validated_data['new_password'])
            user.save()
            return Response({'message': 'Password changed successfully.'}, status=status.HTTP_200_OK)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

○ Chức năng cập nhật thông tin (Cần đăng nhập và xác thực)


```
class UpdateProfileView(APIView):
    permission_classes = [IsAuthenticated]
    authentication_classes = [SafeJWTAuthentication]

    def put(self, request):
        serializer = UpdateProfileSerializer(instance=request.user, data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data, status=status.HTTP_200_OK)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

- Chức năng xác nhận token (Cần xác thực)

```
class VerifyTokenView(APIView):
    authentication_classes = [SafeJWTAuthentication]
    permission_classes = [IsAuthenticated]

    def get(self, request):
        return Response({'message': 'Token is valid.'}, status=status.HTTP_200_OK)
```

- Vào *urls.py* của *user* rồi thêm các đường dẫn:

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/ecomSys/user/register/', RegisterView.as_view(), name='register'),
    path('api/ecomSys/user/login/', LoginView.as_view(), name='login'),
    path('api/ecomSys/user/info/', UserDetailView.as_view(), name='user-detail'),
    path('api/ecomSys/user/change/', ChangePasswordView.as_view(), name='change'),
    path('api/ecomSys/user/update/', UpdateProfileView.as_view(), name='update'),
    path('api/ecomSys/user/verify-token/', VerifyTokenView.as_view(), name='verify-token'),
]
```

2. Manager Service

Tương tự như User Service, Manager Service chỉ có sự khác biệt ở một số điểm sau

- Trong *models.py*, không còn dùng *email* để xác thực người dùng nữa mà thay vào đó sử dụng *username*

```
class Manager(AbstractUser):
    phone_number = models.CharField(max_length=20)
    email = models.EmailField(max_length=255)

    def __str__(self):
        return self.username
```

- Trong *views.py*, việc thêm một Manager khác đòi hỏi phải xác thực.

3. Books Service

- Tạo app *book_service* trong book bằng lệnh *py manage.py book_service*
- Vào *settings.py* để thay đổi một số cấu hình:
 - Thêm `PORT = 4002`

```
ALLOWED_HOSTS = ['localhost']
PORT = 4002
```

- Thêm *corsheaders*, *rest_framework*, *book_service* vào

INSTALLED_APPS

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'corsheaders',  
    'rest_framework',  
    'book_service'  
]
```

- Cấu hình CSDL phù hợp với MongoDB

```
DATABASES = {  
    'default': {  
        'ENGINE': 'djongo',  
        'NAME': 'db_books',  
        'HOST': 'localhost',  
        'PORT': 27017,  
    }  
}
```

- Thêm MEDIA_URL và MEDIA_ROOT để truy cập dữ liệu ảnh

```
MEDIA_URL = '/media/'  
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
```

- Thêm CORS_ALLOW_CREDENTIALS và CORS_ORIGIN_WHITELIST

```
CORS_ALLOW_CREDENTIALS = True # to accept cookies via ajax request  
CORS_ORIGIN_WHITELIST = [  
    'http://localhost:3000' # the domain for front-end app(you can add more than 1)  
]
```

- Thêm đường dẫn sau vào *urls.py* của book để truy cập các dữ liệu media

```
urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

- Vào *models.py* của *book_service* ○ Tạo Category

```
class Category(models.Model):  
    category_id = models.CharField(max_length=7, primary_key=True)  
    name = models.CharField(max_length=100, unique=True)  
    is_active = models.BooleanField(default=True)  
    des = models.TextField(null=True)  
  
    def __str__(self):  
        return self.name
```

- Tạo Book

```
class Book(models.Model):
    book_id = models.CharField(max_length=7, primary_key=True)
    title = models.CharField(max_length=100)
    author = models.CharField(max_length=100)
    image = models.ImageField(upload_to='image/books/')
    price = models.FloatField()
    sale = models.FloatField()
    quantity = models.IntegerField()
    is_active = models.BooleanField(default=True)
    des = models.TextField(null=True)

    category = models.ForeignKey(Category, on_delete=models.CASCADE)

    def __str__(self):
        return self.title
```

- Tạo *serializers.py* ở app *book_service*
 - o Tạo *CategorySerializer* nhằm phục vụ các chức năng liên quan đến category

```
class CategorySerializer(serializers.ModelSerializer):
    class Meta:
        model = Category
        fields = ['category_id', 'name', 'des']

    def destroy(self, instance):
        instance.is_active = False
        instance.save()
        return instance
```

- o Tạo *BookSerializer* nhằm mục đích phục vụ việc lưu và xóa book

```
class BookSerializer(serializers.ModelSerializer):
    category_id = serializers.CharField(write_only=True)

    class Meta:
        model = Book
        fields = ['book_id', 'title', 'author', 'image', 'price', 'sale', 'quantity', 'des', 'category_id']

    def create(self, validated_data):
        category_id = validated_data.pop('category_id', None)
        image = validated_data.pop('image', None)
        request = self.context.get('request')

        if category_id:
            category_instance = Category.objects.filter(is_active__in=[True], category_id=category_id).first()
            if category_instance:
                validated_data['category'] = category_instance
            else:
                raise serializers.ValidationError('Category does not exists')
        return Book.objects.create(image=request.FILES.get('image'), **validated_data)

    def destroy(self, instance):
        instance.is_active = False
        instance.save()
        return instance
```

- o Tạo *BookInfoSerializer* nhằm mục đích phục vụ việc xem thông tin book

```
class BookInfoSerializer(serializers.ModelSerializer):
    class Meta:
        model = Book
        fields = ['book_id', 'title', 'author', 'image', 'price', 'sale', 'quantity', 'des']
```

- o Tạo *UpdateBookSerializer* nhằm mục đích phục vụ việc chỉnh sửa book


```

class UpdateBookSerializer(serializers.ModelSerializer):
    category_id = serializers.CharField(write_only=True)

    class Meta:
        model = Book
        fields = ['image', 'price', 'sale', 'quantity', 'des', 'category_id']

    def update(self, instance, validated_data):
        request = self.context.get('request')

        instance.image = request.FILES.get('image')
        instance.price = validated_data.get('price')
        instance.sale = validated_data.get('sale')
        instance.quantity = validated_data.get('quantity')
        category_id = validated_data.pop('category_id')
        category_instance = Category.objects.filter(is_active__in=[True], category_id=category_id).first()
        if category_instance:
            instance.category = category_instance
        else:
            raise serializers.ValidationError('Category does not exist')
        instance.des = validated_data.get('des')

        instance.save()
        return instance

```

- Vào *views.py* của *book_service* viết logic các chức năng
 - Chức năng thêm category (Cần gọi service manager để xác thực)

```

class CreateCategoryView(APIView):
    def post(self, request):
        token_verification_url = "http://localhost:4001/api/ecomSys/manager/verify-token/"
        headers = {'Authorization': request.headers.get('Authorization')}
        response = requests.get(token_verification_url, headers=headers)

        if response.status_code == 200:
            serializer = CategorySerializer(data=request.data)
            if serializer.is_valid():
                serializer.save()
                return Response(serializer.data, status=status.HTTP_201_CREATED)
            return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
        return Response({'error': 'Invalid token.'}, status=status.HTTP_401_UNAUTHORIZED)

```

- Chức năng hiển thị tất cả category

```

class CategoryListView(APIView):
    def get(self, request):
        categories = Category.objects.filter(is_active__in=[True]).all()
        serializer = CategorySerializer(categories, many=True)
        return Response(serializer.data, status=status.HTTP_200_OK)

```

- Chức năng xóa category (Cần gọi service manager để xác thực)

```

class DeleteCategory(APIView):
    def delete(self, request, category_id):
        token_verification_url = "http://localhost:4001/api/ecomSys/manager/verify-token/"
        headers = {'Authorization': request.headers.get('Authorization')}
        response = requests.get(token_verification_url, headers=headers)

        if response.status_code == 200:
            try:
                category = Category.objects.get(category_id=category_id)
            except Category.DoesNotExist:
                return Response({'error': 'Category not found'}, status=status.HTTP_404_NOT_FOUND)

            serializer = CategorySerializer()
            serializer.destroy(category)

            return Response({'message': 'Category soft deleted'}, status=status.HTTP_204_NO_CONTENT)
        return Response({'error': 'Invalid token.'}, status=status.HTTP_401_UNAUTHORIZED)

```

- Chức năng thêm book (Cần gọi service manager để xác thực)


```
class AddBookView(APIView):
    def post(self, request):
        token_verification_url = "http://localhost:4001/api/ecomSys/manager/verify-token/"
        headers = {'Authorization': request.headers.get('Authorization')}
        response = requests.get(token_verification_url, headers=headers)

        if response.status_code == 200:
            serializer = BookSerializer(data=request.data, context={'request': request})
            if serializer.is_valid():
                serializer.save()
                return Response(serializer.data, status=status.HTTP_201_CREATED)
            return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
        return Response({'error': 'Invalid token.'}, status=status.HTTP_401_UNAUTHORIZED)
```

- Chức năng hiển thị tất cả book

```
class BookListView(APIView):
    def get(self, request):
        books = Book.objects.filter(is_active__in=[True]).all()
        serializer = BookInfoSerializer(books, many=True)
        return Response(serializer.data, status=status.HTTP_200_OK)
```

- Chức năng lọc book theo category

```
class BookListofCategoryView(APIView):
    def get(self, request, category_id):
        books = Book.objects.filter(category_id=category_id, is_active__in=[True])
        serializer = BookInfoSerializer(books, many=True)
        return Response(serializer.data, status=status.HTTP_200_OK)
```

- Chức năng tìm kiếm book bằng keyword

```
class SearchBookListView(APIView):
    def get(self, request, key):
        books = Book.objects.filter(Q(title__icontains=key) | Q(author__icontains=key), is_active__in=[True])
        serializer = BookInfoSerializer(books, many=True)
        return Response(serializer.data, status=status.HTTP_200_OK)
```

- Chức năng xem thông tin 1 book

```
class BookDetailView(APIView):
    def get(self, request, book_id):
        book = Book.objects.filter(book_id=book_id, is_active__in=[True]).first()
        serializer = BookInfoSerializer(book)
        return Response(serializer.data, status=status.HTTP_200_OK)
```

- Chức năng cập nhật thông tin book(Cần gọi service manager để xác thực)

```
class UpdateBookView(APIView):
    def put(self, request, book_id):
        token_verification_url = "http://localhost:4001/api/ecomSys/manager/verify-token/"
        headers = {'Authorization': request.headers.get('Authorization')}
        response = requests.get(token_verification_url, headers=headers)

        if response.status_code == 200:
            try:
                book = Book.objects.get(book_id=book_id)
            except Book.DoesNotExist:
                return Response({'error': 'Book not found'}, status=status.HTTP_404_NOT_FOUND)
            serializer = UpdateBookSerializer(book, data=request.data, context={'request': request})
            if serializer.is_valid():
                serializer.save()
                return Response(serializer.data, status=status.HTTP_200_OK)
            return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
        return Response({'error': 'Invalid token.'}, status=status.HTTP_401_UNAUTHORIZED)
```

- Chức năng xóa book(Cần gọi service manager để xác thực)

```

class DeleteBook(APIView):
    def delete(self, request, book_id):
        token_verification_url = "http://localhost:4001/api/ecomSys/manager/verify-token/"
        headers = {'Authorization': request.headers.get('Authorization')}
        response = requests.get(token_verification_url, headers=headers)

        if response.status_code == 200:
            try:
                book = Book.objects.get(book_id=book_id)
            except Book.DoesNotExist:
                return Response({'error': 'Book not found'}, status=status.HTTP_404_NOT_FOUND)

            serializer = BookSerializer()
            serializer.destroy(book)

            return Response({'message': 'Book soft deleted'}, status=status.HTTP_204_NO_CONTENT)

        return Response({'error': 'Invalid token.'}, status=status.HTTP_401_UNAUTHORIZED)

```

- Vào *urls.py* của *books* và thêm các đường dẫn sau

```

urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/ecomSys/category/add/', CreateCategoryView.as_view()),
    path('api/ecomSys/book/add/', AddBookView.as_view()),
    path('api/ecomSys/category/all/', CategoryListView.as_view()),
    path('api/ecomSys/book/all/', BookListView.as_view()),
    path('api/ecomSys/book/detail/<str:book_id>', BookDetailView.as_view()),
    path('api/ecomSys/book/category/<str:category_id>', BookListofCategoryView.as_view()),
    path('api/ecomSys/book/search/<str:key>', SearchBookListView.as_view()),
    path('api/ecomSys/book/edit/<str:book_id>', UpdateBookView.as_view()),
    path('api/ecomSys/book/delete/<str:book_id>', DeleteBook.as_view()),
    path('api/ecomSys/category/delete/<str:category_id>', DeleteCategory.as_view()),
]

```

4. Search Service

- Tạo app *search_service* trong book bằng lệnh *py manage.py search_service*
- Vào *settings.py* để thay đổi một số cấu hình:
 - o Thêm PORT = 4003

```

ALLOWED_HOSTS = ['localhost']
PORT = 4003

```

- o Thêm *corsheaders*, *rest_framework*, *search_service* vào INSTALLED_APPS

```

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'corsheaders',
    'rest_framework',
    'search_service',
]

```

- o Cấu hình CSDL phù hợp với MongoDB

```
DATABASES = {
    'default': {
        'ENGINE': 'django',
        'NAME': 'db_searchs',
        'HOST': 'localhost',
        'PORT': 27017,
    }
}
```

- Thêm CORS_ALLOW_CREDENTIALS và CORS_ORIGIN_WHITELIST

```
CORS_ALLOW_CREDENTIALS = True # to accept cookies via ajax request
CORS_ORIGIN_WHITELIST = [
    'http://localhost:3000' # the domain for front-end app(you can add more than 1)
]
```

- Vào *models.py* của *search_service* để tạo model Search

```
class Search(models.Model):
    key = models.CharField(max_length=255)
    is_active = models.BooleanField(default=True)
    created_at = models.DateTimeField(auto_now_add=True)
    user_id = models.CharField(max_length=7)

    def __str__(self):
        return self.key
```

- Tạo *serializers.py* ở app *search_service* để tạo SearchSerializer

```
class SearchSerializer(serializers.ModelSerializer):
    class Meta:
        model = Search
        fields = ['key']

    def destroy(self, instance):
        instance.is_active = False
        instance.save()
        return instance
```

- Vào *views.py* của *search_service* viết logic các chức năng
 - Chức năng tìm kiếm sản phẩm: chức năng này sẽ gọi đến tất cả các service liên quan đến sản phẩm(books service, clothers service, mobiles service) để lấy dữ liệu search


```

class SearchView(APIView):
    def post(self, request):
        key = request.query_params.get('key', '')

        token_verification_url = "http://localhost:4000/api/ecomSys/user/info/"
        headers = {'Authorization': request.headers.get('Authorization')}
        response = requests.get(token_verification_url, headers=headers)

        if response.status_code == 200:
            user_id = response.json().get('id')
            Search.objects.create(key=key, user_id=user_id)

        result = []
        result += self.search_book(key)

        return Response(result, status=status.HTTP_200_OK)

    def search_book(self, key):
        book_service_url = "http://localhost:4002/api/ecomSys/book/search/{}/".format(key)

        book_response = requests.get(book_service_url)
        if book_response.status_code == 200:
            return book_response.json()
        return []

```

- Chức năng hiển thị tất cả thông tin search của người dùng(Cần gọi service user để xác thực)

```

class ShowSearchView(APIView):
    def get(self, request):
        token_verification_url = "http://localhost:4000/api/ecomSys/user/info/"
        headers = {'Authorization': request.headers.get('Authorization')}
        response = requests.get(token_verification_url, headers=headers)
        if response.status_code == 200:
            user_id = response.json().get('id')
            searchs_instance = Search.objects.filter(is_active=True, user_id=user_id).all()
            serializer = SearchSerializer(searchs_instance, many=True)
            return Response(serializer.data, status=status.HTTP_200_OK)
        return Response({'error': 'Invalid token.'}, status=status.HTTP_401_UNAUTHORIZED)

```

- Chức năng xóa thông tin search của người dùng(Cần gọi service user để xác thực)

```

class DeleteSearchView(APIView):
    def delete(self, request, key):
        token_verification_url = "http://localhost:4000/api/ecomSys/user/info/"
        headers = {'Authorization': request.headers.get('Authorization')}
        response = requests.get(token_verification_url, headers=headers)

        if response.status_code == 200:
            user_id = response.json().get('id')

            try:
                search = Search.objects.get(user_id=user_id, key=key, is_active=True)
            except Search.DoesNotExist:
                return Response({'error': 'Search not found'}, status=status.HTTP_404_NOT_FOUND)

            serializer = SearchSerializer()
            serializer.destroy(search)

            return Response({'message': 'Search soft deleted'}, status=status.HTTP_204_NO_CONTENT)

        return Response({'error': 'Invalid token.'}, status=status.HTTP_401_UNAUTHORIZED)

```

- Vào *urls.py* của *search* và thêm các đường dẫn sau

```

urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/ecomSys/search', SearchView.as_view()),
    path('api/ecomSys/search/show/', ShowSearchView.as_view()),
    path('api/ecomSys/search/delete/<str:key>/', DeleteSearchView.as_view()),
]

```

5. Cart Service

- Tạo app *cart_service* trong book bằng lệnh *py manage.py cart_service*
- Vào *settings.py* để thay đổi một số cấu hình:

```
ALLOWED_HOSTS = ['localhost']  
PORT = 4004
```

- o Thêm *corsheaders*, *rest_framework*, *cart_service* vào
INSTALLED_APPS

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'corsheaders',  
    'rest_framework',  
    'cart_service',  
]
```

- o Cấu hình CSDL phù hợp với MySQL

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.mysql',  
        'NAME': 'db_carts',  
        'USER': 'cartservice',  
        'PASSWORD': '12345',  
        'HOST': 'localhost',  
        'PORT': '3306'  
    }  
}
```

- o Thêm *CORS_ALLOW_CREDENTIALS* và
CORS_ORIGIN_WHITELIST

```
CORS_ALLOW_CREDENTIALS = True # to accept cookies via ajax request  
CORS_ORIGIN_WHITELIST = [  
    'http://localhost:3000' # the domain for front-end app(you can add more than 1)  
]
```

- Vào *models.py* của *cart_service* để tạo model *CartItem*

```
class CartItem(models.Model):
    user_id = models.CharField(max_length=7)
    date_added = models.DateTimeField(auto_now_add=True)
    quantity = models.IntegerField()
    type = models.CharField(max_length=50)
    product_id = models.CharField(max_length=7)
    is_active = models.BooleanField(default=True)
```

- Tạo *serializers.py* ở app *cart_service*
 - o Tạo *CartItemSerializer* nhằm mục đích phục việc thêm và xóa *CartItem*

```
class CartItemSerializer(serializers.ModelSerializer):
    class Meta:
        model = CartItem
        fields = ['user_id', 'quantity', 'type', 'product_id']

    def destroy(self, instance):
        instance.is_active = False
        instance.save()
        return instance
```

- o Tạo *UpdateCartItemSerializer* nhằm mục đích phục vụ việc cập nhật *CartItem*

```
class UpdateCartItemSerializer(serializers.ModelSerializer):
    class Meta:
        model = CartItem
        fields = ['quantity']

    def update(self, instance, validated_data):
        instance.quantity = validated_data.get('quantity')
        instance.save()
        return instance
```

- Vào *views.py* của *cart_service* viết logic các chức năng
 - o Chức năng thêm sản phẩm vào giỏ (Cần gọi service user để xác thực và lấy dữ liệu người dùng)

```
class AddToCartView(APIView):
    def post(self, request):
        token_verification_url = "http://localhost:4000/api/ecomSys/user/info"
        headers = {'Authorization': request.headers.get('Authorization')}
        response = requests.get(token_verification_url, headers=headers)

        if response.status_code == 200:
            user_id = response.json().get('id')
            product_id = request.data.get('product_id')
            cart_item = CartItem.objects.filter(is_active=True, user_id=user_id, product_id=product_id).first()
            if cart_item:
                serializer = UpdateCartItemSerializer(instance=cart_item, data=request.data)
                if serializer.is_valid():
                    serializer.save()
                    return Response(serializer.data, status=status.HTTP_200_OK)
                return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
            else:
                request.data['user_id'] = user_id
                serializer = CartItemSerializer(data=request.data)
                if serializer.is_valid():
                    serializer.save()
                    return Response(serializer.data, status=status.HTTP_201_CREATED)
                return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
        return Response({'error': 'Invalid token.'}, status=status.HTTP_401_UNAUTHORIZED)
```


- Chức năng xem giỏ hàng: Chức năng này sẽ gọi đến service user để xác thực và lấy dữ liệu người dùng. Từ dữ liệu người dùng, chức năng lấy ra các CartItem và gọi đến service sản phẩm tương ứng với sản phẩm trong CartItem để lấy dữ liệu sản phẩm.

```
class CartView(APIView):
    def get(self, request):
        token_verification_url = "http://localhost:4000/api/ecomSys/user/info"
        headers = {'Authorization': request.headers.get('Authorization')}
        response = requests.get(token_verification_url, headers=headers)

        if response.status_code == 200:
            user_id = response.json().get('id')
            cart_items = CartItem.objects.filter(is_active=True, user_id=user_id)
            cart_total = 0
            cart_item_data = []

            for cart_item in cart_items:
                product = self.get_product(cart_item.type, cart_item.product_id)
                if product:
                    cart_item_data.append({
                        'quantity': cart_item.quantity,
                        'product': product,
                        'total': cart_item.quantity * product.get('price', 0) * (100-product.get('sale', 0))/100
                    })
                    cart_total += cart_item.quantity * product.get('price', 0) * (100-product.get('sale', 0))/100
            response_data = {
                'cart_items': cart_item_data,
                'cart_total': cart_total
            }
            return Response(response_data, status=status.HTTP_200_OK)
        return Response({'error': 'Invalid token.'}, status=status.HTTP_401_UNAUTHORIZED)

    def get_product(self, type, product_id):
        if type == 'book':
            product_url = "http://localhost:4002/api/ecomSys/book/detail/{}/".format(product_id)
            response = requests.get(product_url)
            if response.status_code == 200:
                return response.json()
            return None
```

- Chức năng xóa hàng khỏi giỏ(Cần gọi service user để xác thực và lấy dữ liệu người dùng)

```
class DeleteCartItemView(APIView):
    def delete(self, request, product_id):
        token_verification_url = "http://localhost:4000/api/ecomSys/user/info"
        headers = {'Authorization': request.headers.get('Authorization')}
        response = requests.get(token_verification_url, headers=headers)

        if response.status_code == 200:
            user_id = response.json().get('id')

            try:
                cart_item = CartItem.objects.get(user_id=user_id, product_id=product_id, is_active=True)
            except CartItem.DoesNotExist:
                return Response({'error': 'CartItem not found'}, status=status.HTTP_404_NOT_FOUND)

            serializer = CartItemSerializer()
            serializer.destroy(cart_item)
            return Response({'message': 'CartItem soft deleted'}, status=status.HTTP_204_NO_CONTENT)
        return Response({'error': 'Invalid token.'}, status=status.HTTP_401_UNAUTHORIZED)
```

- Vào *urls.py* của *cart* và thêm các đường dẫn sau

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/ecomSys/cart/add/', AddToCartView.as_view()),
    path('api/ecomSys/cart/show/', CartView.as_view()),
    path('api/ecomSys/cart/delete/<str:product_id>/', DeleteCartItemView.as_view()),
]
```

6. Clothers Service

[Chưa cập nhật]

7. Mobiles Service

[Chưa cập nhật]

8. Order Service

[Chưa cập nhật]

9. Payment Service

[Chưa cập nhật]

10. Shipment Service [Chưa cập nhật]

III. Search, Create Cart, Add To Cart

Dùng Postman để thực hiện việc Rest API cho search, create cart, add to cart.

1. Search

- Khi người dùng chưa đăng nhập sẽ không lưu thông tin search vào cơ sở dữ liệu
 - o Nhập đường dẫn <http://localhost:4003/api/ecomSys/search> và chọn phương thức POST

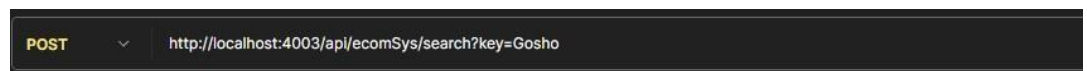


POST http://localhost:4003/api/ecomSys/search?key=Gosho

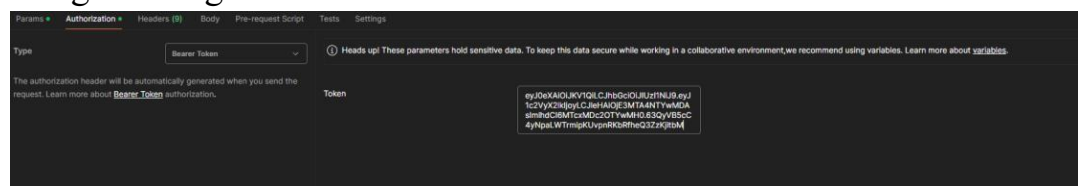
- o Nhấn gửi, kết quả thu được


```
Body Cookies Headers (10) Test Results
Pretty Raw Preview Visualize JSON
[
  {
    "book_id": "CN00001",
    "title": "Thám tử lừng danh Conan tập 1",
    "author": "Gosho AOYAMA",
    "image": "/media/image/books/conan1_2_mE0LkuD.jpg",
    "price": 36000.0,
    "sale": 5.0,
    "quantity": 30,
    "des": "Rất hay và đáng đọc"
  },
  {
    "book_id": "CN00059",
    "title": "Thám tử lừng danh Conan tập 59",
    "author": "Gosho AOYAMA",
    "image": "/media/image/books/conan79_d0iSqgd.jpg",
    "price": 36000.0,
    "sale": 4.0,
    "quantity": 50,
    "des": "Rất hay và đáng đọc"
  },
  {
    "book_id": "CN00072",
    "title": "Thám tử lừng danh Conan tập 72",
    "author": "Gosho AOYAMA",
    "image": "/media/image/books/conan72_s7K6J7d.jpg",
    "price": 38000.0,
    "sale": 6.0,
    "quantity": 60,
    "des": "Hay lắm"
  },
  {
    "book_id": "CN00026",
    "title": "Thám tử lừng danh Conan tập 26",
    "author": "Gosho AOYAMA",
    "image": "/media/image/books/conan26.jpg",
    "price": 32000.0,
    "sale": 4.0,
    "quantity": 50,
  }
]
```

- Khi người dùng đã đăng nhập sẽ lưu thông tin search của người dùng vào cơ sở dữ liệu
 - o Nhập đường dẫn <http://localhost:4003/api/ecomSys/search> và chọn phương thức POST



- o Ở phần *Authorization*, chọn *Bearer Token* và nhập mã token xác thực của người dùng



- o Nhấn gửi, kết quả thu được

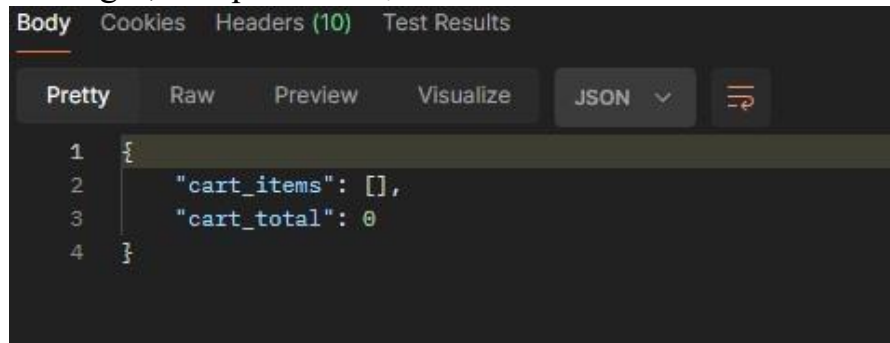
```
Body Cookies Headers (10) Test Results
Pretty Raw Preview Visualize JSON
[
  {
    "book_id": "CN00001",
    "title": "Thám tử lừng danh Conan tập 1",
    "author": "Gosho AOYAMA",
    "image": "/media/image/books/conan1_2_mE0LkuD.jpg",
    "price": 36000.0,
    "sale": 5.0,
    "quantity": 30,
    "des": "Rất hay và đáng đọc"
  },
  {
    "book_id": "CN00059",
    "title": "Thám tử lừng danh Conan tập 59",
    "author": "Gosho AOYAMA",
    "image": "/media/image/books/conan79_d0iSsgd.jpg",
    "price": 36000.0,
    "sale": 4.0,
    "quantity": 50,
    "des": "Rất hay và đáng đọc"
  },
  {
    "book_id": "CN00072",
    "title": "Thám tử lừng danh Conan tập 72",
    "author": "Gosho AOYAMA",
    "image": "/media/image/books/conan72_s7K6J7d.jpg",
    "price": 38000.0,
    "sale": 6.0,
    "quantity": 60,
    "des": "Hay lắm"
  },
  {
    "book_id": "CN00026",
    "title": "Thám tử lừng danh Conan tập 26",
    "author": "Gosho AOYAMA",
    "image": "/media/image/books/conan26.jpg",
    "price": 32000.0,
    "sale": 4.0,
    "quantity": 50,
  }
]
```

- Nhập đường dẫn <http://localhost:4003/api/ecomSys/search/show/>, chọn phương thức GET và vào phần *Authorization*, chọn *Bearer Token*, nhập mã token xác thực của người dùng để xem thông tin search
- Nhấn gửi, kết quả thu được

```
Body Cookies Headers (10) Test Results
Pretty Raw Preview Visualize JSON
[
  {
    "key": "Gosho"
  }
]
```

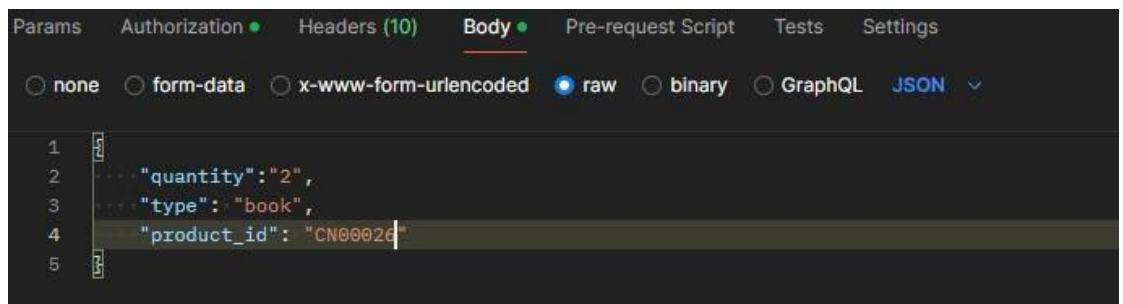
2. Create Cart, Add To Cart

- Nhập đường dẫn <http://localhost:4003/api/ecomSys/cart/show/>, chọn phương thức GET và vào phần *Authorization*, chọn *Bearer Token*, nhập mã token xác thực của người dùng để xem thông tin cart
- Nhấn gửi, kết quả thu được



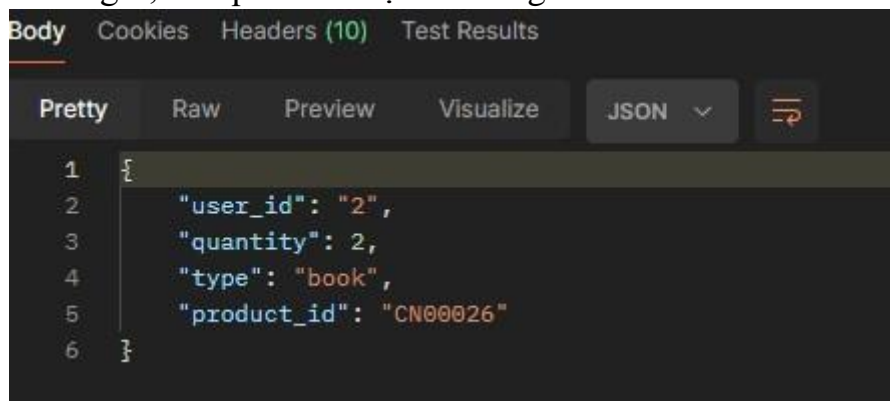
```
Body Cookies Headers (10) Test Results
Pretty Raw Preview Visualize JSON
1 {
2   "cart_items": [],
3   "cart_total": 0
4 }
```

- Nhập đường dẫn <http://localhost:4003/api/ecomSys/cart/add/>, chọn phương thức POST và vào phần *Authorization*, chọn *Bearer Token*, nhập mã token xác thực của người dùng để thêm sản phẩm vào cart
- Ở phần *Body*, chọn *raw* rồi chọn kiểu dữ liệu *Json* rồi nhập dữ liệu của CartItem



```
Params Authorization Headers (10) Body Pre-request Script Tests Settings
none form-data x-www-form-urlencoded raw binary GraphQL JSON
1 {
2   "quantity": "2",
3   "type": "book",
4   "product_id": "CN00026"
5 }
```

- Nhấn gửi, kết quả thu được là thông tin CartItem vừa thêm



```
Body Cookies Headers (10) Test Results
Pretty Raw Preview Visualize JSON
1 {
2   "user_id": "2",
3   "quantity": 2,
4   "type": "book",
5   "product_id": "CN00026"
6 }
```

- Nhập đường dẫn <http://localhost:4003/api/ecomSys/cart/show/>, chọn phương thức GET và vào phần *Authorization*, chọn *Bearer Token*, nhập mã token xác thực của người dùng để xem thông tin cart

- Nhấn gửi, kết quả thu được thông tin giỏ hàng với sản phẩm vừa thêm

```
Body Cookies Headers (10) Test Results
Pretty Raw Preview Visualize JSON ↕
1 {
2   "cart_items": [
3     {
4       "quantity": 2,
5       "product": {
6         "book_id": "CN00026",
7         "title": "Thám tử lừng danh Conan tập 26",
8         "author": "Gosho AOYAMA",
9         "image": "/media/image/books/conan26.jpg",
10        "price": 32000.0,
11        "sale": 4.0,
12        "quantity": 50,
13        "des": "Rất hay và đáng đọc"
14      },
15      "total": 61440.0
16    }
17  ],
18  "cart_total": 61440.0
19 }
```